IMPERIAL

Imperial College London

Faculty of Engineering

Department of Computing

Indiscriminate Data Poisoning Against Supervised Learning: General Attack Formulations, Robust Defences, and Poisonability

Javier Carnerero Cano

Submitted in partial fulfilment of the requirements for the degree of *Doctor of Philosophy* in Computing of the Imperial College London

December 2023

Statement of Originality

I hereby declare that this thesis, titled "Indiscriminate Data Poisoning Against Supervised Learning: General Attack Formulations, Robust Defences, and Poisonability", and the research presented in it are my own, and that all information derived from the work of others is properly acknowledged. I also confirm that this work was done wholly or mainly while in candidature for a research degree at the Imperial College of Science, Technology and Medicine.

Signed: Javier Carnerero Cano

Date: December 2023

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

This research was sponsored by the Defence Science and Technology Laboratory (Dstl) and was accomplished under the project Evaluating the Robustness of Machine Learning Algorithms in Adversarial Settings (ERASE); contract no.: DSTLX-1000120987. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of Dstl. Dstl is authorised to reproduce and distribute reprints notwithstanding any copyright notation here on.

Abstract

MACHINE learning (ML) systems often rely on data collected from untrusted sources, such as humans or sensors, that can be compromised. These scenarios expose ML algorithms to data poisoning attacks, where adversaries manipulate a fraction of the training data to degrade the ML system performance. However, previous works lack a systematic evaluation of attacks considering the ML pipeline, and focus on classification settings. This is concerning since regression models are also applied in safety-critical systems.

We characterise indiscriminate data poisoning attacks and defences in worst-case scenarios against supervised learning algorithms, considering the ML pipeline: data sanitisation, hyperparameter learning, and training.

We propose a novel attack formulation that considers the effect of the attack on the model's hyperparameters. We apply this attack formulation to several ML classifiers using L_2 and L_1 regularisation. Our evaluation shows the benefits of using regularisation to help mitigate poisoning attacks, when hyperparameters are learnt using a trusted dataset.

We then introduce a threat model for poisoning attacks against regression models, and propose a novel stealthy attack formulation via multiobjective bilevel optimisation, where the two objectives are attack effectiveness and detectability. We experimentally show that state-of-the-art defences do not mitigate these stealthy attacks. Furthermore, we theoretically justify the detectability objective and methodology designed. We also propose a novel defence, built upon Bayesian linear regression, that rejects points based on the model's predictive variance. We empirically show its effectiveness to mitigate stealthy attacks and attacks with a large fraction of poisoning points.

Finally, we introduce the concept of "poisonability", which allows us to find the

number of poisoning points required so that the mean error of the clean points matches the mean error of the poisoning points on the poisoned model. This challenges the underlying assumption of most defences. Specifically, we determine the poisonability of linear regression.

Keywords: adversarial machine learning, artificial intelligence, artificial intelligence security, attacks, bilevel optimisation, classification algorithms, data poisoning, deep learning, defences, detectability constraints, hyperparameter optimisation, machine learning, poisonability, regularisation, regression algorithms, reverse-mode differentiation, security, supervised learning.

To my family and friends.

Out of the night that covers me,
Black as the pit from pole to pole,
I thank whatever gods may be
For my unconquerable soul.

In the fell clutch of circumstance,
I have not winced nor cried aloud.
Under the bludgeonings of chance
My head is bloody, but unbowed.

Beyond this place of wrath and tears
Looms but the Horror of the shade,
And yet the menace of the years
Finds, and shall find, me unafraid.

It matters not how strait the gate,
How charged with punishments the scroll,
I am the master of my fate
I am the captain of my soul.

William Ernest Henley, "Invictus".

Everything passes on and everything remains,
But our lot is to pass on,
To go on making paths,
Paths across the sea.

I never sought glory,
Nor to leave my song
In the memory of man;
I love those subtle worlds,
Weightless and graceful,
As bubbles of soap.

I like to watch as they paint themselves
In sunlight and scarlet, floating
Beneath the blue sky, trembling
Suddenly then popping...

I never sought glory.

Traveler, your footprints
Are the path and nothing more;
Traveler, there is no path,
The path is made by walking.

By walking the path is made
And when you look back
You'll see a road
Never to be trodden again.

Traveler, there is no path,
Only trails across the sea...

Some time past in that place
Where today the forests are dressed in barbs
A poet was heard to cry
"Traveler, there is no path,
The path is made by walking..."

Beat by beat, verse by verse...

The poet died far from home.
He lies beneath the dust of a neighboring land.
As he walked away he was seen to weep.
"Traveler, there is no path,
The path is made by walking..."

Beat by beat, verse by verse...

When the goldfinch cannot sing,
When the poet is a pilgrim,
When prayer will do us no good.
"Traveler, there is no path,
The path is made by walking..."

Beat by beat, verse by verse.

Antonio Machado, "Proverbios y Cantares" (Trans.: Asa Cusack).

Acknowledgements

FIVE years of research training at an institution such as the Imperial College London represent an invaluable experience. And what a five years! Added to this, meeting so many talented and brilliant people, life in different countries and cultures, a global pandemic, conflicts around the world, lots of life lessons and personal ups and downs can lead to so many stories to tell, but can also provide a profound personal growth. The contributions written in Ch. 1 are merely academic formalities, but the actual contribution of this thesis and experience is knowing and overcoming myself, even in my darkest days. In this regard, here I want to acknowledge those people who significantly contributed to my research, work and/or personal life throughout these years.

I would like to express my heartfelt gratitude to my supervisor, Prof Emil C. Lupu, for his guidance, insights, unwavering support throughout the course of my PhD journey—including the difficult years of the COVID-19 pandemic—and for ensuring the financial support towards the end of my PhD. His patience, dedication, passion for security research, and critical thinking have been instrumental in shaping my academic and personal growth. In this regard, his ability to encourage independent thinking while providing constructive feedback has empowered me to get out of my comfort zone in terms of thinking, writing, presenting, and outreach. I still remember the old days when I used to explain myself mostly using "Greek" terms—a.k.a. obscure maths.

I would also like to extend my thanks to my (unofficial but practical) second supervisor, Dr Luis Muñoz González, for the insightful discussions of machine learning ideas, for his contributions to our works, for providing feedback that helped improve my mathematical thinking, and for being enjoyable to work with. Moreover, his genuine belief in my potential[1]

[1]Thanks also to Prof Aníbal R. Figueiras Vidal, RIP. It all started with him.

have been pivotal in making this academic endeavour a reality. I am extremely grateful that Luis believed I could handle research in machine learning, which I had not much experience in before my PhD—I should remember I came from the world of antennas, radio frequency, and electromagnetic sensors. By looking at the signal-to-noise ratio of my research, I cannot resist questioning myself whether this precious opportunity was given to right person. This question perhaps cannot be answered with certainty, but in the following years of my career I hope to prove that it might not be a complete mistake.

Working with Emil and Luis has not been a bed of roses, though. In the graduate student descent algorithm of my research, they have always encouraged me to find the global minimum, while I was aiming to find a saddle point, sometimes with stubbornness. These years have been arduous and uncomfortable, but yet interesting, challenging, memorable, and ruthlessly pushing the limit of my patience, knowledge, and sanity.

Many thanks as well to my thesis examiners Prof Ben Glocker and Dr Fabio Pierazzi for their time and valuable feedback.

I gratefully acknowledge financial support for this research from the Defence Science and Technology Laboratory (Dstl), under the project Evaluating the Robustness of Machine Learning Algorithms in Adversarial Settings (ERASE); contract no.: DSTLX-1000120987. This support allowed me to focus on my research and devote time to the intellectual pursuits that culminated in this dissertation. In particular, thanks to Dr Phillippa Spencer OBE for her collaboration, interest, and kindness shown throughout my PhD studies.

I would also like to thank Dr Amani El-Kholy, who was always so helpful and assisted me several times throughout my PhD. Also, thanks to my British "mother" and teacher, Ms Susan Peneycad, for always being willing to help and listen to my miscellaneous problems.

I wish to thank everyone in the Resilient Information Systems Security (RISS) group, for having been great company to share an office, work, and breaks. The exchange of ideas and interdisciplinary perspectives has enriched my academic experience immeasurably. Special thanks to Marwa, Zaid, Aaron, Luca, Kenny, and Jukka for the great chats about research, life, and international food. Particularly, thanks to Marwa for providing such a refreshing and lively atmosphere inside and outside the university, for the good and difficult moments shared together, and for being there when support was needed. *Shukran.*

I would also like to thank the undergraduate, master, and PhD students I worked with, in particular: Greg, Matteo, Bjarne, Paul, and Patrick.

It is very easy to feel isolated as a PhD student. However, during my PhD, I was lucky enough to meet some people whose support helped me a lot in my personal life. Therefore, I would like to thank for all the shared experiences, coffees, conversations, good times, and adventures with: Ibrahim, Borja, Shibei, Alexandros, Lyssa, Henrik, Harleen, Raúl, Mehran, Peyman, and others. Ibrahim: when I first arrived in London, you were the first who genuinely gave me a hand and stood by my side. Eternal gratitude.

I would also like to thank IBM Research Europe - Ireland for the opportunity they provided me to carry out my research in a cutting-edge industrial lab, and the wonderful colleagues and collaborators I shared my time with: Giulio, Ambrish, Zaid, Mark, Juan, and others. I cannot forget other interns: Furkan, Federica, Nick, Leopold, and Leon: thank you for your support, the experiences and the beers we shared together in Dublin. Also, thanks to my flatmates in Dublin, Rafa and Pepe, for sharing support and good moments, and for dealing with our dearest landlord.

Because we stand on the shoulders of giants, I will always be in debt to my bachelor's and masters' ($\times 2$) theses supervisor, Dr Francisco Javier Herraiz Martínez for instilling me his passion about science, for giving me the confidence to become "something of a scientist" myself, and for being such a brilliant mentor, researcher, and person. Thanks also to Dr Julio Ramiro Bargueño for providing support during some difficult times of my PhD.

Last but not least, thanks to my lifelong friends in Madrid, including Escudero, Vallejo, and Félix, and to my family, for their unconditional support throughout my studies and their patience when I was sometimes absent. Félix, Escudero, Pablo, David, and Patri: I will never forget that surprise visit in London (Brexit Day) that cheered me up when I was not in my best moment. Vallejo and Gabe: thanks for that wonderful *Camino*. I am also grateful to my sister Palomi who always believed in me and supported me on countless times. Especially, I want to express my gratitude towards my parents Paloma and Modesto for their unconditional love, support, visits, and concern for my well-being. To a large extent, I am the person who I am because of them.

To all of you: *thank you*.

Javier Carnerero Cano

Contents

Abstract iii

Acknowledgements vii

Mathematical Notation xvii

List of Figures xix

List of Tables xxviii

List of Algorithms xxx

List of Acronyms xxxii

1 Introduction 1

1.1 Motivation . . . 3

1.2 Research Gaps and Research Questions . . . 5

1.2.1 Research Gaps . . . 5

1.2.2 Research Questions . . . 6

1.3 Thesis Statement, Goals and Methodology . . . 7

1.3.1 Thesis Statement and Goals . . . 7

1.3.2 Methodology . . . 8

1.4 Thesis Scope . . . 9

1.5 Thesis Contributions and their Significance . . . 11

1.6 Publications . . . 16

1.7 Thesis Outline . . . 17

2 Background on Data Poisoning and Bilevel Optimisation 21

2.1 Machine Learning . . . 21
2.1.1 Supervised Learning . . . 22
2.2 Classical Assumptions for Machine Learning and Security Implications . . . 24
2.3 Adversarial Machine Learning . . . 26
2.3.1 The Importance of Adversarial Machine Learning . . . 26
2.3.2 Threat Model . . . 27
2.4 Data Poisoning . . . 34
2.4.1 Data Poisoning Attacks Algorithms . . . 37
2.4.2 Optimal Poisoning Attacks . . . 42
2.4.3 Defences Against Data Poisoning . . . 44
2.5 Bilevel Optimisation . . . 48
2.6 Conclusions . . . 54

3 Hyperparameter Learning under Data Poisoning 55
3.1 Introduction . . . 55
3.2 Related Work . . . 57
3.3 General Optimal Poisoning Attacks . . . 58
3.3.1 Problem Formulation . . . 58
3.3.2 Solving General Optimal Poisoning Attacks . . . 62
3.3.3 Projected Hypergradient Descent/Ascent . . . 67
3.4 Regularisation to Partially Mitigate Poisoning Attacks . . . 68
3.5 Experiments . . . 71
3.5.1 Experimental Settings . . . 71
3.5.2 Logistic Regression . . . 74
3.5.3 Deep Neural Networks . . . 79
3.6 Conclusions . . . 85

4 Regression Poisoning: Threat Model 87
4.1 Introduction . . . 87
4.2 Regression Formulation and Classical Assumptions . . . 88
4.3 Threat Model . . . 91
4.3.1 Attacker's Capability . . . 91
4.3.2 Attacker's Goals . . . 94
4.3.3 Error-Specific Attacks . . . 98

4.3.4 Attacker's Knowledge . . . 103
4.4 On the Difficulty of Poisoning Regression Models . . . 104
4.5 Conclusions . . . 106

5 Stealthy Attacks Against Regression and Defences 109
5.1 Introduction . . . 109
5.2 Related Work . . . 112
5.3 Attack with Soft Detectability Constraints . . . 114
5.3.1 Optimal Attack Formulation . . . 114
5.3.2 Optimal Attack Formulation with Soft Detectability Constraints . . . 116
5.3.3 Solving Optimal Poisoning Attacks with Soft Detectability Constraints 123
5.4 BayesClean: A Defence Based on Uncertainty . . . 127
5.4.1 Bayesian Linear Regression . . . 128
5.4.2 Predictive Variance under Data Poisoning . . . 131
5.4.3 BayesClean . . . 131
5.5 Experimental Evaluation . . . 134
5.5.1 Experimental Settings . . . 135
5.5.2 Attacks with Detectability Constraints . . . 138
5.5.3 BayesClean . . . 143
5.6 Conclusions . . . 146

6 Analysis of Optimal Indiscriminate Attacks Against Ridge Regression 149
6.1 Introduction . . . 149
6.2 Analysis of the Convexity of the Attacker's Objective . . . 151
6.2.1 Convexity in the Poisoning Labels . . . 153
6.2.2 Convexity in the Basis Functions . . . 154
6.3 Most of the Critical Points of the Attacker's Objective Function are Saddle Points . . . 159
6.4 Asymptotic Behaviour of the Attacker's Objective Function . . . 161
6.4.1 Asymptotic Behaviour in the Poisoning Labels . . . 162
6.4.2 Asymptotic Behaviour in the Basis Functions . . . 162
6.4.3 Over-Regularised Ridge Regression under Data Poisoning . . . 164
6.5 Equivalence Between the Optimal Indiscriminate Attack and Concave Quadratic Programming, If Only the Labels Are Optimised . . . 167

6.5.1 On the Number of Combinations of Poisoning Labels 169
6.5.2 Bounds on the Rank of the Hyper-Hessian Matrix and No. of Poisoning Points Optimised Simultaneously 170
6.6 Soft Detectability Constraints Do Not Preserve (in General) the Convexity of the Attacker's Objective 173
6.6.1 Example 1: Constraining the Perturbations of the Poisoning Labels with a Poisoned Model as a Reference 174
6.6.2 Example 2: Constraining the Perturbations of the Poisoning Labels with a Clean Model as a Reference 176
6.6.3 Example 3: Constraining the Perturbations of the Model's Parameters: Machine Teaching 176
6.6.4 Upper Bound on the Rank of the Hyper-Hessian when Considering Detectability Constraints 177
6.6.5 Eigenvalues of the Hyper-Hessian Matrix and Convexity 178
6.7 Challenges of the Optimal Attack Formulation with Soft Detectability Constraints 179
6.7.1 Second-Order Challenge: Curvature 180
6.7.2 First-Order Challenge: Attacker's KKT Conditions and Diversity of Poisoning Points 180
6.7.3 Zeroth-Order Challenge: Order of Magnitude of Objective Functions 186
6.7.4 Proposed Detectability-Risk Function 186
6.8 Conclusions 188

7 Poisonability of Regression Models 191

7.1 Introduction 191
7.2 Related Work 192
7.3 On the Poisonability of Models and Datasets 193
7.3.1 Formulation of the Problem 195
7.3.2 Solving the Poisonability Problem 195
7.3.3 Approximation to the Problem 197
7.4 On the Poisonability of Linear Regression 198
7.4.1 Non-Regularised Linear Regression 200
7.4.2 Ridge Regression 206

7.4.3 Discussion on Possible Extensions 210
7.5 On the Degree of the Polynomial for Ridge Regression 213
7.5.1 Characterising the Polynomial in the Objective Function 214
7.5.2 Upper Bound on the Degree of the Polynomial 215
7.6 Conclusions 216

8 Conclusions **217**
8.1 Summary of Thesis Achievements and their Implications 217
8.2 Future Work 222

Bibliography **227**

A Woodbury Formula, Sherman-Morrison Formula, and Matrix Determinant Lemma **257**
A.1 Woodbury Formula 257
A.2 Sherman-Morrison Formula and Matrix Determinant Lemma 258

B Inverse of Block Matrix Whose Diagonal Blocks Are Non-Singular **259**

C Bounds on Bilinear Quadratic Forms **261**

D Robustness of Update of Positive Definite and Positive Semidefinite Matrices **263**

E On the Rank of Matrices **265**
E.1 Rank of Scatter and Gram Matrices 265
E.2 Multiplication by a Non-Singular Matrix Preserves Rank 266
E.3 Upper Bound on the Rank of the Multiplication of Two Matrices 267
E.4 Upper Bound on the Rank of the Sum of Two Matrices 267

Mathematical Notation

We use a notation similar to (Bishop, 2006; Bishop and Bishop, 2024). Objective functions are denoted by uppercase calligraphic letters, such as $\mathcal{A}$. Datasets and models are denoted by $\mathcal{D}$ and $\mathcal{M}$, respectively. Clean training, clean validation, clean test, target, and poisoning datasets are denoted by $\mathcal{D}_{\text{tr}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{test}}$, $\mathcal{D}_{\text{target}}$, and $\mathcal{D}_{\text{p}}$, respectively. The complete training dataset is denoted by the union of the clean training and poisoning sets, $\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}$.

Input and output spaces are denoted by $\mathcal{X}$ and $\mathcal{Y}$, respectively. Unless otherwise stated, unidimensional variables and constants are denoted by lower case italic letters such as x. Vectors are denoted by lower case bold letters such as $\mathbf{x}$, whereas matrices are denoted by uppercase bold letters, such as $\mathbf{X}$. All vectors are assumed to be column vectors unless otherwise stated. A superscript $\star$ denotes an optimal solution to an optimisation problem, e.g., $\mathbf{x}^\star$. A superscript T denotes the transpose of a matrix or vector, so that $\mathbf{x}^\mathsf{T}$ will be a row vector. The notation $[x_1 \ \ldots \ x_M]$ denotes a row vector with M elements, and the corresponding column vector is written as $\mathbf{x} = [x_1 \ \ldots \ x_M]^\mathsf{T}$. The $M \times M$ identity matrix is denoted $\mathbf{I}_M$, which will be abbreviated to $\mathbf{I}$ if there is no ambiguity about its dimensionality. $\mathbf{1}_M$ and $\mathbf{0}_M$ denote M-dimensional column vectors in which all elements have the value 1 and value 0, respectively—also abbreviated to $\mathbf{1}$ and $\mathbf{0}$, respectively, if there is no ambiguity about their dimensionality. $\mathbf{1}_{M\times D}$ and $\mathbf{0}_{M\times D}$ denote $M \times D$ matrices in which all elements have the value 1 and value 0, respectively. $|x|$ denotes the absolute value of a scalar x. $\text{sgn}(x)$ denotes the sign function of x. $||\mathbf{x}||_2$ denotes the ℓ_2-norm of a vector $\mathbf{x}$, a.k.a. the 2-norm or Euclidean norm. $||\mathbf{x}||_1$, $||\mathbf{x}||_\infty$, and $||\mathbf{x}||_p$ denote the ℓ_1-norm (a.k.a. Manhattan norm), ℓ_∞-norm (a.k.a. maximum norm or Chebyshev norm), and ℓ_p-norm (a.k.a. Minkowski norm), respectively, of $\mathbf{x}$. We use $\det(\mathbf{A})$ and $\text{rank}(\mathbf{A})$ to denote the determinant and rank of a matrix $\mathbf{A}$, respectively.

We use $\mathbb{R}$, $\mathbb{R}_{>a}$, $\mathbb{R}_{\geq a}$, $\mathbb{Z}$, $\mathbb{Z}_c$, and $\mathbb{N}$ to denote the real numbers, real numbers greater than a, real numbers greater than or equal to a, integer numbers, c-ary integer numbers (c possible values), and natural numbers, respectively. Curly braces $\{\}$ denote a set. $|\mathcal{D}|$ denotes the size of the set $\mathcal{D}$. $\text{dom}(f(\mathbf{x}))$ denotes the domain of $f(\mathbf{x})$. $\log f(\mathbf{x})$ and $\exp(f(\mathbf{x}))$ denote the natural logarithm and exponential functions, respectively, of $f(\mathbf{x})$. $\nabla_{\mathbf{x}} f(\mathbf{x})$, $\partial_{\mathbf{x}} f(\mathbf{x})$, $\nabla_{\mathbf{x}} \nabla_{\mathbf{x}} f(\mathbf{x})$, and $\nabla^2_{\mathbf{x}} f(\mathbf{x})$ denote gradient, subdifferential, Hessian matrix, and Hessian matrix of $f(\mathbf{x})$ with respect to $\mathbf{x}$, respectively. $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ and $\frac{df(\mathbf{x})}{d\mathbf{x}}$ denote partial derivative and total derivative of $f(\mathbf{x})$ with respect to $\mathbf{x}$, respectively. The notation $g(x) = \mathcal{O}(f(x))$ denotes that $|f(x)/g(x)|$ is bounded as $x \to \infty$. The notation $\lceil x \rceil$ denotes the "ceiling" of x, i.e., the smallest integer that is greater than or equal to x.

The notation $\mathbf{x} \sim p(\mathbf{x})$ signifies that the random vector $\mathbf{x}$ is sampled from the distribution $p(\mathbf{x})$. The expectation of a function $f(\mathbf{x}, \mathbf{y})$ with respect to a random variable $\mathbf{x}$ is denoted by $\mathbb{E}_{\mathbf{x}}[f(\mathbf{x}, \mathbf{y})]$. In situations where there is no ambiguity as to which variable is being averaged over, this will be simplified by omitting the suffix, for instance $\mathbb{E}[\mathbf{x}]$. Similarly, the variance of $f(x)$ is denoted $\text{Var}[f(x)]$, and for vector variables, the covariance is written $\text{Cov}[\mathbf{x}, \mathbf{y}]$. We will also use $\text{Cov}[\mathbf{x}]$ as a shorthand notation for $\text{Cov}[\mathbf{x}, \mathbf{x}]$.

A Gaussian or normal distribution with mean μ and variance σ^2 is denoted $\mathcal{N}(x|\mu, \sigma^2)$ or, if there is no ambiguity, $\mathcal{N}(\mu, \sigma^2)$. Similarly, a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is denoted $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or, if there is no ambiguity, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. A continuous uniform distribution on the interval $[a, b]$ is denoted $\mathcal{U}(x|a, b)$ or, if there is no ambiguity, $\mathcal{U}(a, b)$. If we have N independent and identically distributed values $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of a D-dimensional vector $\mathbf{x} = [x_1 \ \ldots \ x_D]^\mathsf{T}$, we can combine the observations into a data matrix (a.k.a. design matrix) $\mathbf{X}$ of dimension $N \times D$ in which the nth row of $\mathbf{X}$ corresponds to the row vector $\mathbf{x}_n^\mathsf{T}$. Thus, the n, i element of $\mathbf{X}$ corresponds to the ith element of the nth observation $\mathbf{x}_n$ and is written $x_{n,i}$ or, equivalently, $[\mathbf{X}]_{n,i}$.

Other mathematical notations are defined when they first appear in the corresponding chapter.

List of Figures

1.1 Main contributions of this thesis in terms of the main players in data poisoning: threat models, attacks, defences, and evaluation of data poisoning attacks. 11

1.2 Thesis outline. Each arrow implies that the corresponding previous chapter is needed to understand the current chapter. We also include the research questions (RQs) and contributions (Cs) of each chapter. 18

2.1 Synthetic binary classification problem. The red and green points represent the instances for each class, and the solid black curves some possible decision boundary. **(a)** The curve is underfitting. **(b)** The plot describes the trend of the data reasonably well. **(c)** The decision boundary is overfitting (Troy, 2016). 24

2.2 Illustration of security threats and defensive techniques of ML (Liu et al., 2018a). 28

2.3 Synthetic example of data poisoning in a binary classification problem. The points denote training examples, and the colours denote distinct classes. Starred points indicate poisoning points intentionally positioned to distort the decision boundary. The red line denotes the learnt decision boundary and the blue line, the true decision boundary. **(a)** Effect on the decision boundary after injecting one poisoning point. **(b)** Effect after injecting five poisoning points (Muñoz-González and Lupu, 2019). 36

2.4 Preventing exploitation of face data using error-minimising noise. In this scenario, the defender wants to prevent their face images from being exploited to train face recognition or verification systems. The user has access to a small dataset of facial images and uses this small dataset to generate and apply the error-minimising noise to their own face images before sharing them on online social media platforms. After this, the unlearnable version of face images on social media is collected to train a DNN-based facial recognition or verification system. The system is then used to recognise one of the user's clean face images captured somewhere by a camera. The goal is to prevent the DNN from learning the defender's face images and make it perform poorly on their clean face images (Huang et al., 2021). 41

2.5 Data poisoning and anomaly detection. **(a)** In non-adversarial settings, the defender can often learn the model's parameters that (approximately) fit the clean training data. **(b)** The addition of poisoning data can significantly change the learnt parameters, leading to high test error. **(c)** By discarding outliers and then training on the remaining data, the defender can mitigate the effectiveness of the attacker. In this example, the defender discards all blue points outside the blue ellipse, and all red points outside the red ellipse (Koh et al., 2022). 44

3.1 Effect of regularisation on a synthetic example. The blue and green points represent the training data points for each class, and the red point is the poisoning point (labelled as green). The dashed-dotted grey box represents the attacker's constraints. The dashed-white lines and solid-red lines depict the decision boundaries for LR classifiers trained on clean and poisoned datasets respectively. (a) Standard LR with no regularisation. (b) LR with L_2 regularisation. The colourmaps in the two plots represent the validation error as a function of the poisoning point. (c) Value of λ learnt by solving Eq. (3.1) as a function of the location of the injected poisoning point. . . . 69

3.2 Average test error for the optimal attack against LR: The first row represents the case of L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. The second row contains the plots for L_1 regularisation on (d) MNIST, (e) FMNIST, and (f) CIFAR-10. 74

3.3 Average λ and $||\mathbf{w}||_2^2$ for the optimal attack against LR on MNIST: (a) L_2 regularisation; (b) L_1 regularisation. 75
3.4 Average λ and average $||\mathbf{w}||_2^2$ for the optimal attack against LR: (a) L_2 regularisation on FMNIST, (b) L_2 regularisation on CIFAR-10, (c) L_1 regularisation on FMNIST, and (d) L_1 regularisation on CIFAR-10. 76
3.5 Sensitivity analysis of the size of the validation set. Results for (a) and (d) MNIST, (b) and (e) FMNIST, and (c) and (f) CIFAR-10. (a)—(c) Results when there is no regularisation. (b)—(f) Results for L_2 regularisation. . . . 77
3.6 Sensitivity analysis of the size of the validation set for L_1 regularisation: (a) MNIST, (b) FMNIST, and (c) and CIFAR-10. 77
3.7 Average Kuncheva's consistency index for the optimal attack against LR using L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. . . 79
3.8 Average Kuncheva's consistency index for the optimal attack against LR, using L_2 regularisation (first and second rows); and using L_1 regularisation (third to fifth rows), on MNIST (first column), FMNIST (second column) and CIFAR-10 (third column), for different numbers of selected features, k: (a) $k = 40$, (b) $k = 80$, (c) $k = 400$, (d) $k = 80$, (e) $k = 160$, (f) $k = 800$, (g) $k = 20$, (h) $k = 40$, (i) $k = 200$, (j) $k = 40$, (k) $k = 80$, (l) $k = 400$, (m) $k = 80$, (n) $k = 160$, and (o) $k = 800$. 80
3.9 Average test error for the optimal attack against the DNNs using L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. 82
3.10 Average test error for the optimal attack against the DNNs using L_1 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. 82
3.11 Average λ learnt with RMD at each layer of the DNNs, using L_2 regularisation, on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. 83
3.12 Average λ learnt with RMD for the DNNs using L_1 regularisation with a different regularisation hyperparameter at each layer. The left, central and right figures represent the results for MNIST, FMNIST and CIFAR-10, respectively. 83

3.13 Average $||\mathbf{w}_{\text{layer}}||_2^2/d_{\text{layer}}$ for the DNNs using L_2 regularisation (first row) and L_1 regularisation (second row) with a different regularisation hyperparameter at each layer, where d_{layer} represents the number of parameters of the corresponding layer. This normalisation allows comparing $||\mathbf{w}_{\text{layer}}||_2^2$ for each layer regardless of the number of the parameters. The first, second and third columns represent the results for MNIST, FMNIST and CIFAR-10, respectively. 84

3.14 Average λ and average $||\mathbf{w}||_2^2/d$ for the optimal attack against the DNNs using a single regularisation hyperparameter, where d is the number of parameters of the model. The first row represents the case of L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. The second row represents the case of L_1 regularisation on (d) MNIST, (e) FMNIST, and (f) CIFAR-10. . 84

4.1 Synthetic example of an indiscriminate error-generic attack targeting the bias term of the model. The target model is linear regression. The blue points represent the clean training points, the green points represent the poisoning points, and the two red points represent a poisoning point at the beginning and at the end of its optimisation process, respectively, whose trajectory during the optimisation process is depicted by the magenta line. The colourmap represents the negative of the value of the attacker's objective, which aims to minimise the bias term of the model. The white dashed line is the regression line learnt under the clean data, the red line—which is translated w.r.t. the white line—represents the regression line learnt under the poisoning data, and grey bounding box represents the constraints of the poisoning points. Certainly, locating poisoning points in the yellow region can decrease the bias term of the model. 96

4.2 Possible error-specific attacks in regression settings. In the diagram, the black arrow denote the access point for the attacker in the ML pipeline. The red arrows denote the goals of the attacker. Each stage of the cycle can affect the next stages. Evaluation means that the model performance is checked before deployment. Deployment means "test" time in the real world, where we have previously unseen data. When monitoring the model, unexpected changes can lead to a redefinition of the problem and assumptions. 99

4.3 Example of a possible heteroscedastic attack inspired by the concept of amplitude modulation in signal processing. In this example, the genuine data ("carrier") is multiplied by an adversarial noise that varies through the input space. The result is the poisoned data, whose mean is similar and whose residual variance changes through the input space. For instance, this form of noise can lead to new forms of detectability constraints for the attacker. 100

4.4 Uncertainty in depth estimation. The first image is an example input into a Bayesian model which estimates depth, as shown by the second image. The third image shows the estimated uncertainty. We can observe that the model predicts the wrong depth on difficult surfaces, such as the red car's reflective and transparent windows. Thankfully, the Bayesian model is also aware it is wrong and exhibits increased uncertainty (Kendall, 2017). 102

5.1 Synthetic example showing the effect of constraining poisoning points against TRIM (Jagielski et al., 2018). The training dataset contains 37 clean points and 13 poisoning points (26% of the training set). The features of the clean points are generated according to a uniform distribution $\mathcal{U}(-5, 5)$, the labels are generated according to the transformation: $y = 0.8x$, and the additive noise of the labels follows a Gaussian distribution $\mathcal{N}(\epsilon|0.0, 1.2^2)$. The linear regression model and TRIM are trained using SGD with a learning rate $\eta = 0.1$ and # epochs $= 40$ (full batch). The blue points represent the clean points, the green points are the poisoning points, the red crosses are the points rejected by TRIM, the black dashed line is the regression line learnt under the original clean data, the green dotted line is the regression line learnt under the complete poisoned training data, and the red solid line is the regression line learnt under the training data not rejected by TRIM. Because the poisoning points are closer to the clean points, TRIM fails to detect them. 118

5.2 Effect of not normalising the objective functions in a synthetic example. The blue points represent the clean points, the green points are the poisoning points, and the two red points represent a poisoning point at the beginning and at the end of its optimisation process, respectively. Its trajectory during the optimisation process is depicted by the magenta line. The white dashed line is the regression line learnt under the original clean data, and the red solid line is the regression line learnt under the complete poisoned training data. The colourmap represents the value of the attacker's objective as a function of the location of the poisoning point. The experimental settings are the same as in Fig. 5.1. (a) $\alpha = 1$. (b) $\alpha = 0.6$. (c) $\alpha = 0.4$. (d) $\alpha = 0.2$. Note the sudden transition for the poisoning points from outliers ($\alpha = 1$) to inliers ($\alpha = \{0.6, 0.4, 0.2\}$). This is because the detectability objective dominates the effectiveness objective. 120

5.3 Effect of normalising the objective functions in a synthetic example. The blue points represent the clean points, the green points are the poisoning points, and the two red points represent a poisoning point at the beginning and at the end of its optimisation process, respectively. Its trajectory during the optimisation process is shown by the magenta line. The white dashed line is the regression line learnt under the original clean data, and the red solid line is the regression line learnt under the complete poisoned training data. The colourmap represents the value of the attacker's objective as a function of the location of the poisoning point. The experimental settings are the same as in Fig. 5.1. (a) $\alpha = 1$. (b) $\alpha = 0.6$. (c) $\alpha = 0.4$. (d) $\alpha = 0.2$. Compared to Fig. 5.2, in this case, normalising the objective functions allows a smoother transition for the poisoning points from outliers ($\alpha = 1$) to inliers ($\alpha = \{0.6, 0.4, 0.2\}$). 124

5.4 Synthetic example showing the effect of constraining poisoning points against TRIM (Jagielski et al., 2018), for $\alpha = 1$ and $\alpha = 0.4$. The blue points represent the clean points, the green points are the poisoning points, the red crosses are the points rejected by TRIM, the black dashed line is the regression line learned under the original clean data, the green dotted line is the regression line learned under the complete poisoned training data, and the red solid line is the regression line learned under the training data not rejected by TRIM. TRIM fails to detect the stealthy poisoning points, because they are closer to the clean distribution. 125

5.5 Behaviour of BayesClean in a synthetic example. In this example, $c_1 = 1$, $c_2 = 2$, $\lambda_1 = \lambda_2 = \beta_1 = \beta_2 = 10^{-6}$, and $T_{\text{EM}} = 300$ (maximum number of iterations when learning the hyperparameters). The blue points represent the clean points, the green points are the poisoning points, and the green dotted line is the regression line learnt under the complete poisoned training data. The purple region corresponds to the condition $\mathcal{I}_1 = \{i \mid |y_i| \leq |\mu_{*_i} + c_1\sigma_{*_i}|\}$, the red region corresponds to $\mathcal{I}_2 = \{i \mid |\mu_{*_i} + c_1\sigma_{*_i}| < |y_i| \leq |\mu_{*_i} + c_2\sigma_{*_i}|\}$. (a) 2% of poisoning points, and the white region corresponds to $\mathcal{I}_3 = \{i \mid |y_i| > |\mu_{*_i} + c_2\sigma_{*_i}|\}$. (b) 10% of poisoning points. (c) 18% of poisoning points. The predictive variance clearly increases with the number of poisoning points. 133

5.6 Comparison of BayesClean and TRIM in a synthetic example. The experimental settings are the same as in Fig. 5.1 and Fig. 5.5. The blue points represent the clean points, the green points are the poisoning points, the red crosses are the points rejected by the corresponding defence, the black dashed line is the regression line learnt under the original clean data, the green dotted line is the regression line learnt under the complete poisoned training data, and the red solid line is the regression line learnt under the training data not rejected by the corresponding defence. (a) BayesClean, 20% of poisoning points. (b) BayesClean, 24% of poisoning points. (c) BayesClean, 30% of poisoning points. (d) TRIM, 20% of poisoning points. (e) TRIM, 24% of poisoning points. (f) TRIM, 30% of poisoning points. We can observe that BayesClean is more robust than TRIM for large ratios of poisoning points. 133

5.7 Test NMSE of linear regression when there is no defence deployed, for $\alpha = 1$, $\alpha = 0.5$, $\alpha = 0.3$, and $\alpha = 0.1$. (a) Loan. (b) Heart Disease. (c) Boston Housing. (d) Appliances. 139

5.8 Test defence gain of linear regression when using TRIM, Huber, and SEVER. The first, second, third, and fourth rows correspond to Loan, Heart Disease, Boston Housing, and Appliances, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly. 140

5.9 Test NMSE of linear regression on Boston Housing datasets with label noise with different standard deviations (standard Boston Housing, σ, and 2σ), when there is no defence deployed, for $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$. 141

5.10 Test defence gain of linear regression on Boston Housing dataset, for different levels of label noise, when using when using TRIM, Huber, and SEVER. The first and second rows correspond to σ and 2σ, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly. 142

5.11 Test NMSE of the DNNs when there is no defence deployed, for $\alpha = 1$, $\alpha = 0.5$, $\alpha = 0.3$, and $\alpha = 0.1$. (a) Loan. (b) Heart Disease. 142

5.12 Test defence gain of the DNNs when using TRIM, Huber, and SEVER. The first and second rows correspond to Loan and Heart Disease, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly. 143

5.13 Test defence gain of linear regression when using BayesClean, compared to state-of-the-art defences. The first, second, third, and fourth rows correspond to Loan, Heart Disease, Boston Housing, and Appliances, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly. 144

5.14 Test defence gain of the DNNs when using BayesClean, compared to state-of-the-art defences. The first row represents the results for Loan, whereas the second row corresponds to the results for Heart Disease. On the other hand, the first, second, and third columns correspond to the results for $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, respectively. 145

8.1 Main achievements of this thesis considering the ML pipeline. 223

List of Tables

1.1 Thesis contributions: indiscriminate poisoning attacks and defences in the ML pipeline. 12

3.1 Complexity of state-of-the-art methods to compute the hypergradient for the poisoning points and hyperparameters, assuming that SGD is the solver for the inner problem. 65
3.2 Characteristics of the datasets used in the experiments. 71
3.3 Experimental settings for the poisoning attack. 72
3.4 Experimental settings for testing the attacks. 72

4.1 Possible attacker's objectives depending on the specificity of the error and the targeted samples. 95

5.1 Characteristics of the datasets used in the experiments. 135
5.2 Experimental settings for the poisoning attack. 136
5.3 Experimental settings for testing the attacks. 137

List of Algorithms

3.1 RMD for computing the hypergradients w.r.t. the hyperparameters and poisoning points . . . 66
3.2 Projected hypergradient descent/ascent for learning the hyperparameters and poisoning points . . . 67

5.1 Computation of B_{p} poisoning points ($\alpha = 1$) and of the normalisation coefficient for the detectability-risk function . . . 122
5.2 Computation of B_{p} poisoning points ($\alpha \neq 1$) and of the normalisation coefficient for the effectiveness function . . . 123
5.3 RMD for multiobjective optimisation in regression poisoning . . . 126
5.4 Projected hypergradient ascent for regression poisoning . . . 127
5.5 BayesClean . . . 131

List of Acronyms

ADALINE	**ADA**ptive **LI**near **NE**uron/**ADA**ptive **LIN**ear **E**lement
ANR	**A**ttack-to-**N**oise **R**atio
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
BLPP	**Bi**-**L**evel **P**rogramming **P**roblem
C	**C**ontribution
CG	**C**onjugate **G**radient
CIFAR	**C**anadian **I**nstitute **F**or **A**dvanced **R**esearch
CQP	**C**oncave **Q**uadratic **P**rogramming
DNN	**D**eep **N**eural **N**etwork
DP-SGD	**D**ifferentially-**P**rivate **S**tochastic **G**radient **D**escent
DUTI	**D**ebugging **U**sing **T**rusted **I**tems
EM	**E**xpectation-**M**aximisation
FAIL	**F**eatures, **A**lgorithms, **I**nstances, and **L**everage
FMD	**F**orward-**M**ode **D**ifferentiation
FMNIST	**F**ashion **MNIST**
GAN	**G**enerative **A**dversarial **N**etwork
GP	**G**aussian **P**rocess
GPR	**G**aussian **P**rocess **R**egression
GPT	**G**enerative **P**re-trained **T**ransformer
IID	**I**ndependent and **I**dentically **D**istributed
IoT	**I**nternet **o**f **T**hings
KKT	**K**arush-**K**uhn-**T**ucker
LID	**L**ocal **I**ntrinsic **D**imensionality

LK	**L**imited **K**nowledge
LR	**L**ogistic **R**egression
ML	**M**achine **L**earning
MNIST	**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
MSE	**M**ean **S**quared **E**rror
NMSE	**N**ormalised **M**ean **S**quared **E**rror
PK	**P**erfect **K**nowledge
ReLU	**Re**ctified **L**inear **U**nit
RKHS	**R**eproducing **K**ernel **H**ilbert **S**pace
RL	**R**einforcement **L**earning
RMD	**R**everse-**M**ode **D**ifferentiation
RMSE	**R**oot-**M**ean-**S**quared **E**rror
RONI	**R**eject **O**n **N**egative **I**mpact
RQ	**R**esearch **Q**uestion
RVM	**R**elevance **V**ector **M**achine
SD	**S**urrogate **D**ata
SGD	**S**tochastic **G**radient **D**escent
SL	**S**urrogate **L**earner
SVD	**S**ingular **V**alue **D**ecomposition
SVM	**S**upport **V**ector **M**achine
ZK	**Z**ero **K**nowledge

Chapter 1

Introduction

Synopsis

Data poisoning attacks represent a concern for machine learning researchers and practitioners. We characterise indiscriminate data poisoning attacks and defences in worst-case scenarios against supervised learning. In summary, we propose:

- A novel optimal attack formulation that considers the interactions between the attack and the model's hyperparameters.
- Learning the optimal regularisation hyperparameter, using a small trusted set, as a practical methodology to help mitigate poisoning attacks and obtain the best trade-off between accuracy and robustness.
- A novel threat model of poisoning attacks against regression models.
- A novel optimal attack formulation capable of modelling different trade-offs between effectiveness and detectability in regression settings.
- A novel defence, based on the model uncertainty, to mitigate strong and stealthy attacks in regression settings.
- A theoretical analysis of optimal attacks against ridge regression models, showing the challenges and limitations of the attack formulations presented.
- A characterisation of the "poisonability" of linear regression: we determine the number of poisoning points needed so that the error of the poisoning points matches the error of the clean points on the poisoned model.

MACHINE learning (ML) has demonstrated outstanding performance as a general tool in many application domains, including self-driving vehicles, object recognition, sentiment analysis, medical diagnosis, time series forecasting, and chatbots. Motivated by the growing number, refinement and variability of recent attacks and security threats, it is also an appealing solution for security-sensitive applications like malware clustering, intrusion detection, network traffic analysis, or spam filtering (Chio and Freeman, 2018). Although it has been shown that these algorithms accomplish human-level accuracy or better on a number of benchmark tasks, they are not inherently designed to deal with the adversarial nature of some problems (McDaniel et al., 2016; Muñoz-González and Lupu, 2018).

Indeed, several real-world attacks against systems like anti-virus engines, spam filters, or fake profile detection, have already been reported (Muñoz-González and Lupu, 2018, 2019). In another example, an attacker could crash a self-driving car by making the car vision system misclassify a manipulated stop sign as a speed limit sign (Eykholt et al., 2018). In this sense, ML algorithms are often susceptible to specific input perturbations carefully crafted (at training and/or test time) that can cause wrong predictions, and thus compromise the data processing system. Formally, this can be modelled as a violation of the independent and identically distributed (IID) assumption for the dataset (Barreno et al., 2010; Dalvi et al., 2004). Beyond security sensitive contexts, the commercial use of learning algorithms in advertising, social media, and recommender systems provides attackers with new opportunities to compromise the system.

Unfortunately, our understanding of the nature and magnitude of the vulnerabilities of ML models is still incomplete. This has raised concerns about the application of ML in sensitive domains (McDaniel et al., 2016). Hence, given that ML is being pervasively deployed in critical systems that encourage malicious adversaries to tamper with it, it is crucial to understand, manage and assure the behaviour of these systems. In this regard, adversarial ML is a research area that lies at the intersection of ML and security, and aims to analyse the security of ML algorithms (Huang et al., 2011). Concretely, it intends to: (i) identify the attack surface and vulnerabilities of the system; (ii) design the corresponding effective attacks and evaluate their impact; and (iii) propose defences that can help these algorithms to be robust to adversaries (Biggio et al., 2014).

In particular, depending upon the influence the attacker has on the dataset, we can

distinguish between *poisoning* (or causative) attacks, in which an attacker can subvert the learning process by injecting malicious data into the training dataset; and *evasion* (or exploratory) attacks, which are produced at test time and aim at producing intentional errors or evading the system (Barreno et al., 2006, 2010; Huang et al., 2011; Biggio et al., 2014; Muñoz-González and Lupu, 2019). On the other hand, we can also distinguish indiscriminate attacks—which aim to produce errors for a broad set of samples—targeted attacks (Muñoz-González and Lupu, 2019)—which aim to produce errors for reduced set of samples—and subpopulation attacks (Jagielski et al., 2021)—which affect a subset of samples that share certain characteristics.

1.1 Motivation

Unlike traditional software engineering, where suitable quality assurance methods such as code reviews and model-based testing are available, learning models usually lack a structure that supports their manual review, and comprehensive specifications for exhaustive testing are rarely available (Ehlers et al., 2019). For example, a typical coverage metric is the number of lines executed by a verification program. This metric does not apply correctly to ML because it does not reflect how much of the model's logic has been verified, as many computations involved are defined as matrix operations (Papernot, 2018). On this matter, challenges often encountered in ML—e.g., exhaustively enumerating all points near which the classifier should present the same behaviour, or finding the right abstractions for representing components of a neural network's logic (Papernot, 2018)—are still making this goal particularly difficult to achieve (Huang et al., 2017b; Katz et al., 2017; Pulina and Tacchella, 2010). Another reasonable action plan to evaluate the impact of an attack on a particular learning system is providing worst-case guarantees: attack strategies that benefit attackers the most when they have perfect knowledge about the system (Papernot, 2018).

Relevance of data poisoning: many different attacks can be staged against ML models. In particular, evasion attacks—specifically in computer vision applications (Papernot et al., 2018)—have caught the attention of the vast majority of the research community, since they are technically less complex to craft and evaluate because the target model is already trained and therefore fixed. However, a recent survey (Kumar et al., 2020) shows that poisoning is the top perceived threat against ML in industry. Data poisoning can

happen when training data is collected from untrusted sources, e.g., sensors, devices, people, etc. Data can then be manipulated in a such a way to subvert the learning process and achieve a malicious goal, for instance to increase the error of the model. Indeed, several sources confirm that poisoning is already carried out in practice (Kumar et al., 2020; Grosse et al., 2023). For instance, Google declared in 2008: "In practice, we regularly see some of the most advanced spammer groups trying to throw the Gmail filter off-track by reporting massive amounts of spam emails as not spam..." (Constantin, 2021). Another real-world situation involves Google's VirusTotal scanning service, which many antivirus vendors use to augment their own data. While attackers have been known to test their malware against VirusTotal before deploying it in the wild, thereby evading detection, they can also use it to engage in more persistent poisoning. In fact, in 2015 there were reports that intentional sample poisoning attacks through VirusTotal were performed to cause antivirus vendors to detect benign files as malicious (Constantin, 2021). In another example, Microsoft's chatbot Tay was designed to learn language by interacting with users on Twitter, but instead learnt offensive statements (Neff and Nagy, 2016). More recently, Carlini et al. (2023) showed that poisoning web-scale datasets—such as Wikipedia snapshots—is possible under realistic assumptions.

The previous examples teach us that, as learning systems are increasingly being deployed in real-world applications, it is critical to ensure that these systems are trustworthy. Therefore, aspects relating to the security and robustness of ML at training time are of utmost importance for the development of the theory, design, and applications of learning systems. For these reasons, we consider the problem of data poisoning an important problem for both the ML and security communities, and we expect that its relevance will grow even further during the next few years.

Optimal attacks as a practical tool to test robustness: attempting to prove a lower bound on robustness guarantees is still conservative and considerably difficult to put into practice, requiring approximations (Steinhardt et al., 2017; Ma et al., 2019; Wang et al., 2020; Rosenfeld et al., 2020; Chen et al., 2022; Jia et al., 2022). In particular, specifically crafted attacks can degrade the robustness guarantees of certifiably robust models (Mehra et al., 2021). Moreover, robustness is not guaranteed in certain applications where the dataset distribution evolves through time, e.g., malware detection. In contrast, crafting optimal attacks can help estimate an upper bound in a more pragmatic manner,

determining that the ML system meets the security requirements (Goodfellow et al., 2014b; Xiao et al., 2015; Mei and Zhu, 2015; Papernot et al., 2016b; Muñoz-González et al., 2017).

1.2 Research Gaps and Research Questions

1.2.1 Research Gaps

In the context described above, in this thesis we shall focus on worst-case data poisoning attacks. We observed two fundamental gaps in the literature:

- **Data poisoning considering the ML pipeline**: although effective attacks have been proposed against a variety of models, we noticed that there is a lack of systematic methodologies for security evaluation of learning algorithms against indiscriminate poisoning attacks considering the ML pipeline. For example, the Features, Algorithms, Instances, and Leverage (FAIL) framework of Suciu et al. (2018) analyses ML attacks in settings with a variable amount of adversarial knowledge and control over the victim, but they mainly focus on stealthy targeted poisoning attacks. Moreover, they consider hyperparameters constant regardless of the strength of the attack, and disregard the importance of a trusted validation set to provide both accuracy and robustness regardless of the presence or absence of poisoning points in the training dataset. In contrast, we focus on worst-case indiscriminate data poisoning attacks considering data preprocessing/sanitisation, hyperparameter selection, and model training. Hence, understanding the impact of poisoning attacks and their corresponding countermeasures at this level will help design more robust models and relate this impact to the characteristics of the datasets, which has been to a certain extent overlooked in the literature.

- **Data poisoning against regression models**: most of works in the literature focusses on classification settings. We believe that analysing the robustness of regression algorithms is important because regression models are already being applied in safety-critical systems such as medicine for the development of pharmaceuticals (Ekins et al., 2019), hedge funds management (Meligkotsidou et al., 2009), and industrial processes for predictive maintenance (Wen et al., 2022) and quality control (Sun and Hou, 2017). Furthermore, compared to classification, the labels are continuous, which leads to a broad range of attacks depending on the transformation applied

in the label, i.e., a generalisation of attacks in classification settings (in terms of the attacker's objective). Also, it possible to define new detectability constraints considering the poisoning labels.

1.2.2 Research Questions

Having the ML pipeline in mind, we formulate a series of research questions (RQs) that we consider relevant, both at the ML pipeline level, and at the regression learning level.

Data Sanitisation

- **RQ1**: Can we craft attacks with detectability constraints capable of modelling different trade-offs between effectiveness and detectability, and bypassing current defences?
- **RQ2**: Can we design effective defences capable of mitigating strong and stealthy attacks?

Hyperparameter Selection

- **RQ3**: Can we understand the impact of poisoning attacks on the hyperparameters of the model? In particular, can we understand the different trade-offs between accuracy and robustness according to the value of the hyperparameters?
- **RQ4**: Can a small trusted validation set help learn the correct hyperparameters under data poisoning?
- **RQ5**: Can regularisation help mitigate poisoning attacks?

Training Time

- **RQ6**: Can we define how "poisonable" a particular dataset and model are? Specifically, does the assumption that the error of the poisoning points is larger than the error of the clean points always hold true?

Poisoning Attacks Against Regression

- **RQ7**: What are the vulnerabilities of regression models under data poisoning attacks?
- **RQ8**: Can we have a thorough theoretical understanding of optimal indiscriminate attacks against linear regression models?

1.3 Thesis Statement, Goals and Methodology

1.3.1 Thesis Statement and Goals

Within the context previously introduced, our Thesis Statement is as follows:

Thesis Statement

We propose novel indiscriminate data poisoning attacks and defences in worst-cases scenarios against supervised learning algorithms, considering the different stages of the ML pipeline: data prepropressing/sanitisation, hyperparameter selection, and model training. In classification settings, we introduce a novel attack formulation that considers the effect of the attack in the hyperparameters of the model, and show that regularisation, together with a small trusted set, can help mitigate poisoning attacks and obtain the best trade-off between accuracy and robustness. We also expand the threat model of data poisoning attacks against regression models, and introduce and theoretically analyse novel attacks that consider detectability constraints to bypass current defences. We propose a novel defence, based on the uncertainty of the model, capable of mitigating stealthy attacks and attacks that present a large fraction of poisoning points. Moreover, we challenge the underlying assumption of most defences and introduce the concept of "poisonability" of models and datasets: for linear regression models, we theoretically determine the number of poisoning points needed so that the mean error of the poisoning points matches the mean error of the clean points on the poisoned model.

This will be achieved through the lens of:

- **Attacker's characterisation**:
  - Analysis of the influence of the attack in the model's hyperparameters.
  - Analysis of the trade-off between effectiveness and detectability through novel detectability constraints.

- **Defender's characterisation**:
  - Analysis of the trade-off between accuracy and robustness through the learning of the optimal regularisation hyperparameter using a trusted validation set.
  - Measures that can help mitigate stealthy attacks and attacks with a large fraction of poisoning points: uncertainty of the model.
- **Evaluating the impact of data poisoning in models and datasets through their "poisonability"**.
- **Attacks against regression models**:
  - Analysis of new threats in regression settings.
  - Theoretical analysis of optimal indiscriminate attacks.

1.3.2 Methodology

Given the previous statement, it is possible to derive a worst-case analysis to compare the robustness of the algorithms to different levels of data poisoning. This analysis is helpful to build the reliability argument for such systems and determine the maximum damage that an adversary can perpetrate.

The enemy knows the system: we shall follow Kerckhoffs's principle (Kerckhoffs, 1883), "the enemy knows the system", a.k.a. Shannon's maxim (Shannon, 1949), "one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them". This contrasts with other methodologies based on partial knowledge, where there is no standard definition of the attacker's knowledge and often rely on arbitrary choices. We argue that only ML models that have been published and thoroughly analysed—including strong attackers—should be trusted. Otherwise, aspects of the ML model that have been kept secret from the attacker—or, broadly speaking, from the general public—can create a potential failure point.

Evaluating worst-case attacks through bilevel optimisation: bilevel optimisation techniques are mathematical programs, where an optimisation problem (outer level or leader) contains another optimisation problem as a constraint (inner level or follower). Thus, the solution or decision taken by the upper level authority to optimise their goals is

affected by the response of lower level entities, who will seek to optimise their own outcomes (Sinha et al., 2018). These techniques will be developed in the scope of this thesis, because they appear in a natural way in data poisoning problems, as they are able to dynamically model the interactions of attacker and defender (each one aims to maximise their interests) by means of the problem constraints. Furthermore, bilevel programs are also capable of modelling detectability constraints that can be expected from smart attackers, who may want to remain undetected.

Moreover, we aim to make use of this tool to study the implications of the relationship between robustness and accuracy that arises in these problems, to be aware of which scenarios are more suitable to apply a particular defence strategy. For example, implementing an effective defence can increase the robustness of the algorithm to a certain attack strength, but this in turn may worsen its performance on genuine inputs.

Solving bilevel optimisation problems through reverse-mode differentiation: computing optimal attacks is complex, and there is no general method suited to all classes of learning algorithms. Thus, in this project we shall follow reverse-mode differentiation (RMD) (Domke, 2012; Maclaurin et al., 2015; Muñoz-González et al., 2017; Franceschi et al., 2017, 2018; Grazzi et al., 2020) to craft such attacks, which has been proven as a successful and efficient technique to compute the gradients of the attacker.

1.4 Thesis Scope

- **Offline learning settings**: we limit our scope to offline learning—the training dataset is not updated during training—where the attacker can manipulate the training data before the training stage.
- **Adversarial settings**: we are mainly interested in characterising the security and robustness of ML systems at training time, i.e., poisoning attacks. In particular, even though a noisy environment is sometimes called an "adversarial" environment, this thesis focusses specifically on purposefully crafted attacks.
- **Data poisoning**: in this thesis we focus on data poisoning attacks, where the attacker can only manipulate the training data. Consequently, model poisoning attacks and backdoor/Trojan attacks are out of the scope of our research. Model poisoning takes place when an attacker can directly manipulate the parameters,

hyperparameters, gradients, loss function and/or optimisation algorithm (Fang et al., 2020). Backdoor/Trojan attacks, on the other hand, occur when the attacker inserts a trigger in training samples with attacker-specified target labels. Thus, the trained model performs well on benign test samples; however, if the same trigger used in training is added onto a test sample, then its prediction is be changed to the target label specified by the attacker (Liu et al., 2018b; Gu et al., 2019; Wang et al., 2019).

- **Training end-to-end from scratch**: we focus on attacks that are able to poison models end-to-end (i.e., the whole architecture) when training from scratch, in contrast to fine-tuning or transfer learning.

- **Attack strength**: an all powerful attack that can compromise *all* the training points can entirely emulate a possible event occurrence so cannot be detected or defeated. Therefore, we consider attacks that can compromise a variable fraction of the training points. This fraction contributes to characterise the *strength of the attack*, the other aspect being the data perturbations possible.

- **Data and model characteristics**: based on approaches for requirements assurance in traditional safety-related software (Banks and Ashmore, 2019), we restrict ourselves to the requirements that data and model should be syntactically—in terms of the range and constraints of data—and semantically—in terms of the model stability—correct, and address normal and robust behaviour.

- **Defences**: we are mainly interested in defences against data poisoning from two angles: data sanitisation and model robustness. In the case of data sanitisation, we shall explore techniques that detect and reject potentially malicious training points. In consequence, defences that fix/suggest fixes for the training data are out the scope of this thesis.

- **Target models**: we aim to test our methodology in differentiable *supervised learning* models,[1] in particular linear models (logistic regression and linear regression) and deep (fully connected) neural networks, given their common use in the literature and in practice.

- **Hyperparameters analysed**: we focus our study on the regularisation hyperparameters corresponding to L_2 and L_1 regularisation, given their common use by ML

[1] Models in which a gradient-based optimisation can be directly applied.

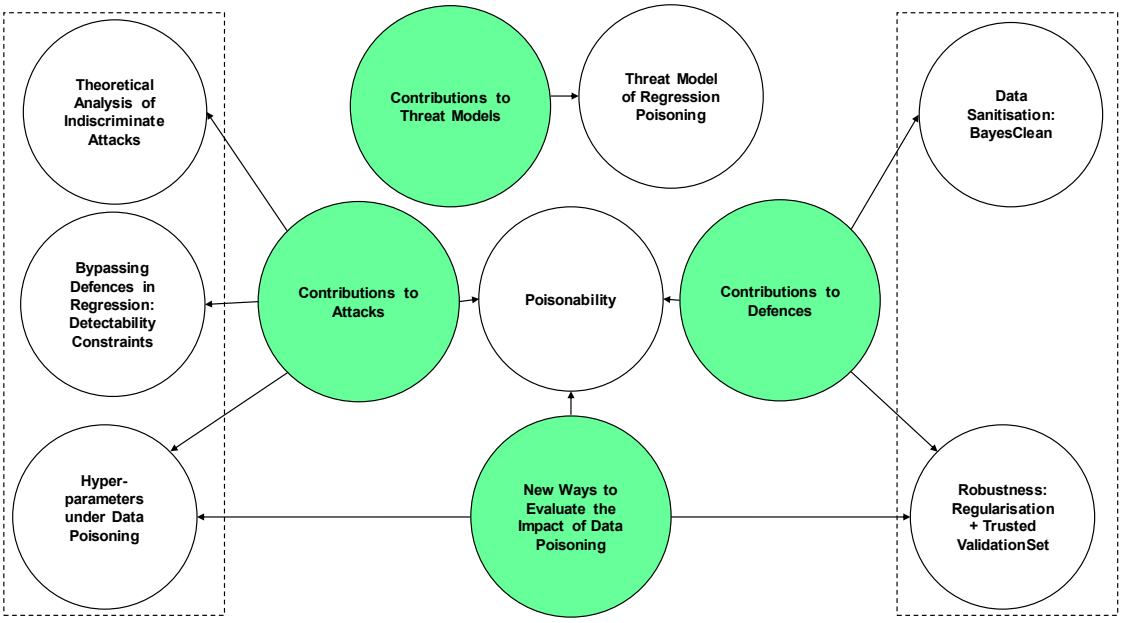


Figure 1.1: Main contributions of this thesis in terms of the main players in data poisoning: threat models, attacks, defences, and evaluation of data poisoning attacks.

scientists and practitioners as a constraint in the model's parameters. Regularisation can help alleviate poisoning attacks and is also useful for our purposes of studying the trade-off between accuracy and robustness.

- **Applications**: the general trend in the field has mainly focussed on computer vision, which defines adversaries using formulations that only make sense in the image domain. This makes it easier for humans to identify and understand malicious manipulations applied to images. Unlike this, we shall consider general tabular data, including not only image classification but also other applications such as house price, interest rate of loans, heart disease, and appliances energy prediction.

1.5 Thesis Contributions and their Significance

We can state the contributions developed throughout this thesis from two different perspectives. The first one is illustrated in Fig. 1.1 in terms of the main players in data poisoning: threat models, attacks, defences, and evaluation of data poisoning attacks. The second one is presented in terms of the ML pipeline in Table 1.1. We also explain the consequences and significance of our contributions to the research community.

<table>
<tr><td rowspan="2">Attacks</td><td>Novel Detectability Constraints</td><td>Hyperparameters under Data Poisoning</td><td rowspan="3">Poisonability</td></tr>
<tr><td colspan="2">Analysis of Optimal Indiscriminate Attacks</td></tr>
<tr><td>Defences</td><td>BayesClean</td><td>Regularisation and Trusted Validation Set</td></tr>
<tr><td></td><td>Data Preprocessing and Sanitisation</td><td>Model and Hyperparameter Selection</td><td>Model Training</td></tr>
</table>

Table 1.1: Thesis contributions: indiscriminate poisoning attacks and defences in the ML pipeline.

- **Contributions to Threat Models**:
  - **C1**: **Threat model of regression poisoning**: we present a large variety of error-specific attacks, including, for instance, assumptions violation, wrong model selection, interpretability violation, and uncertainty fluctuation. Moreover, we theoretically show that in linear regression models, any manipulation of a poisoning point directly affects all the parameters of the model. We also theoretically show that the error caused by a poisoned dataset—when the poisoning points remain undetected by the defender—increases with the noise level of the dataset and with the mismatch between the estimated and the true mapping function of the model.
    - **Significance**: poisoning attacks in regression settings have a much broader range of attack objectives—compared to classification models—and thus a broader spectrum of attacks can be attempted. Combined with their widespread use, this makes regression models a prime and vulnerable target for poisoning attacks. Moreover, our theoretical result implies a limitation of the possibility of creating very targeted attacks or backdoors in linear models, since one of the requirements of a backdoor attack is that the model should behave correctly under clean samples. The result on the poisoning error suggests that datasets that are more difficult to fit to are also more difficult to defend under poisoning.
- **Contributions to Attacks**:
  - **C2**: **General optimal attack formulation considering hyperparameters**: previous attack formulations provide an overly pessimistic view of the robustness to poisoning, since they consider the hyperparameters fixed. We propose a novel

multiobjective bilevel optimisation formulation that accounts for the interactions between the model's hyperparameters and the poisoning attack.

  - **Significance**: this contribution can open new research directions towards attacks that target the hyperparameters and/or the selection of a particular model. For example, a relatively less explored area is architecture search under data poisoning.

- **C3**: **Novel detectability constraints to evade defences**: most of previous works in regression settings propose optimal attacks without considering detectability constraints. These attacks can be easily detected by state-of-the-art defences. We propose a novel optimal attack formulation based on a multiobjective bilevel optimisation problem where, in the outer objective, we introduce detectability constraints. In particular, we model the trade-off between effectiveness and detectability by scalarising two conflicting objectives: effectiveness objective (mean error of the model on clean samples) and detectability risk objective (which penalises poisoning points that are far away from a defined function around the clean points). We also propose a novel detectability-risk function and a novel normalisation algorithm capable of controlling the trade-off between the two objectives.

  - **Significance**: we expect that this contribution will be of interest to both the adversarial ML and the ML optimisation communities. In the adversarial ML community, the lessons learnt evidence the need for specific defences in the regression domain, and defences that also consider stealthy attackers—poisoning points that are close to the clean distribution, which is a challenging problem since it is often difficult to distinguish between genuine and malicious samples in those cases. Moreover, attacks that consider the continuous manipulation of the poisoning labels can lead to new attack formulations with detectability with different goals in mind: they represent a generalisation—in terms of the attacker's objective—from a discrete mapping of the labels (label flipping in classification settings) to a continuous mapping in regression settings. In the ML optimisation community, we technically contribute to solving multiobjective bilevel optimisation problems and bring to light some of its challenges when the

objectives are of different orders of magnitude.

- **C4**: **Theoretical analysis of indiscriminate attacks against ridge regression**: most of the studies in the optimal attacks literature are empirical and do not provide a thorough analysis of the attacker's objective function and its convexity. In contrast, we characterise the attacker's objective function in optimal indiscriminate attacks against ridge regression. This includes the analysis of its convexity, asymptotic behaviour, critical points (namely stationary points), combinations of poisoning labels, equivalent formulations, influence of detectability constraints, and challenges in the optimisation process.
  - **Significance**: this analysis will help design and have a better control over objective functions according to the given specifications—e.g., diversity of poisoning points—and also to illustrate the challenges and possible solutions that can improve the optimisation process.

- **Contributions to Defences**:
  - **C5**: **Model robustness—regularisation and use of a trusted validation set**: regularisation is a well-known technique that can help avoid overfitting, by shrinking the model's parameters. Using a small trusted validation set that serves as a reference for the underlying clean distribution, we show that learning the correct value of the regularisation hyperparameter can provide the best trade-off between accuracy and robustness: a low value of the regularisation hyperparameter provides accuracy when there are not poisoning points, and a larger value can help mitigate the effect of the attack when the number of poisoning points is large.
    - **Significance**: the lessons learnt can help develop ML models more robust to attacks by considering optimal hyperparameters and architectures. This can also open new research directions towards designing models that are robust to data poisoning.
  - **C6**: **BayesClean**: we propose a novel defence against poisoning attacks in regression settings built upon a Bayesian linear regression model. This defence is capable of rejecting poisoning points based on the model's predictive variance under the attack. This technique improves on previous defences from two angles:

(i) it is able to detect poisoning points that consider detectability constraints; and (ii) it more effective than previous defences when the number of poisoning points is large, e.g., more than 15% of the training set.

  - ∗ **Significance**: in contrast to previous defences looking at the errors or the norm of the gradients of the training samples, this shows a promising research direction considering the model uncertainty, and the need to develop techniques that account for this property towards more robust models.

- **New Ways to Evaluate the Impact of Data Poisoning**:

  - **C2**: **General optimal formulation considering hyperparameters**: our novel formulation allows us to provide a more realistic analysis of the impact of the poisoning attack in worst-case scenarios, since the model's hyperparameters can affect the performance of the attack.

    - ∗ **Significance**: this is an important step towards understanding the trade-off between accuracy and robustness to an attack by selecting an appropriate value of the regularisation hyperparameter. In the future, other hyperparameters such as the dropout rate of DNNs can also be explored to understand this trade-off.

  - **C6**: **Model robustness—regularisation and use of a trusted validation set**: this contribution also shows the importance of using a trusted validation set, representative of the clean distribution, to learn the optimal hyperparameters.

    - ∗ **Significance**: although acquiring a trusted set can be expensive and often require human inspection, this investment can pay off if most of the training data is collected from untrusted sources. Therefore, a trusted set is important to guide the learning process of the hyperparameters: if this set is poisoned, the attacker can also corrupt the hyperparameters learnt. There is still room for investigating the best uses of a trusted validation set: both for data sanitisation and for model and hyperparameter selection.

  - **C7**: **Poisonability of datasets and models**: for linear regression, we introduce the concept of poisonability, which allows us to find the number of poisoning points needed so that the mean error of the clean points is larger than the tolerable mean error of the poisoning points on a poisoned model.

* **Significance**: to our knowledge, the concept of poisonability is new in the data poisoning literature. Our definitions and analysis can help understand the consequences of attacks, explain why some defences looking at the error of the training data fail, and define standards of performance degradation with respect to a clean reference model.

1.6 Publications

Parts of the work presented here have been included in the following papers:

- Muñoz-González, L., **Carnerero-Cano, J.**, Co, K. T., and Lupu, E. C. (2019a). Challenges and Advances in Adversarial Machine Learning. In *Resilience and Hybrid Threats: Security and Integrity for the Digital World*, volume 55, page 102. IOS Press.

- **Carnerero-Cano, J.**, Muñoz-González, L., Spencer, P., and Lupu, E. C. (2020). Regularisation Can Mitigate Poisoning Attacks: A Novel Analysis Based on Multiobjective Bilevel Optimisation. *arXiv preprint arXiv:2003.00040*. **Best Poster Award, Machine Learning Summer School Indonesia 2020**.

- **Carnerero-Cano, J.**, Muñoz-González, L., Spencer, P., and Lupu, E. C. (2021). Regularization Can Help Mitigate Poisoning Attacks... with the Right Hyperparameters. In *ICLR Workshop on Security and Safety in Machine Learning Systems*.

- **Carnerero-Cano, J.**, Muñoz-González, L., Spencer, P., and Lupu, E. C. (2023). Hyperparameter Learning under Data Poisoning: Analysis of the Influence of Regularization via Multiobjective Bilevel Optimization. *IEEE Transactions on Neural Networks and Learning Systems*.

- **Carnerero-Cano, J.**, Muñoz-González, L., Spencer, P., and Lupu, E. C. Stealthy Poisoning Attacks Against Regression and Defenses. *Under submission*.

- **Carnerero-Cano, J.**, et al. Poisonability of Ridge Regression. *In preparation*.

Moreover, a side project in data poisoning attacks developed in parallel to this thesis—but not included in it—is presented in the following paper:

- Muñoz-González, L., Pfitzner, B., Russo, M., **Carnerero-Cano, J.**, and Lupu, E. C. (2019b). Poisoning Attacks with Generative Adversarial Nets. *arXiv preprint arXiv:1906.07773*.

1.7 Thesis Outline

We summarise the structure of this thesis in Fig. 1.2. We also include in Fig. 1.2 the corresponding research questions and contributions of each chapter.

Background on data poisoning and bilevel optimisation: in Ch. 2, a theoretical groundwork of supervised learning, intuitions on the vulnerabilities of learning systems, and a review of the threat model in adversarial ML and data poisoning are outlined. Furthermore, bilevel optimisation techniques are briefly described.

Hyperparameter learning under data poisoning: Ch. 3 introduces a novel optimal attack formulation capable of capturing the interactions between the poisoning attack and the model's hyperparameters. We also show that regularisation can help mitigate poisoning attacks, and that learning the regularisation hyperparameter using a trusted validation set provides accuracy when there is no attack, and robustness under the attack.

Threat model of poisoning attacks in regression settings: Ch. 4 introduces a novel threat model for poisoning attacks against regression models. We present a large variety of error-specific attacks, including, for instance, assumptions violation, wrong model selection, interpretability violation, and uncertainty fluctuation. Moreover, we theoretically show that in linear regression models, any manipulation of a poisoning point directly affects all the parameters of the model. This result limits the possibility of creating very targeted attacks or backdoors in linear models. We also theoretically show that the error caused by a poisoned dataset increases with the noise level of the dataset.

Stealthy attacks against regression models and defences: Ch. 5 presents novel attacks against regression models that consider different degrees of detectability constraints to bypass state-of-the-art defences. We also propose a new methodology based on normalisation of objectives to solve the problem and obtain different trade-offs between effectiveness and detectability. We also develop a defence—called BayesClean—built upon

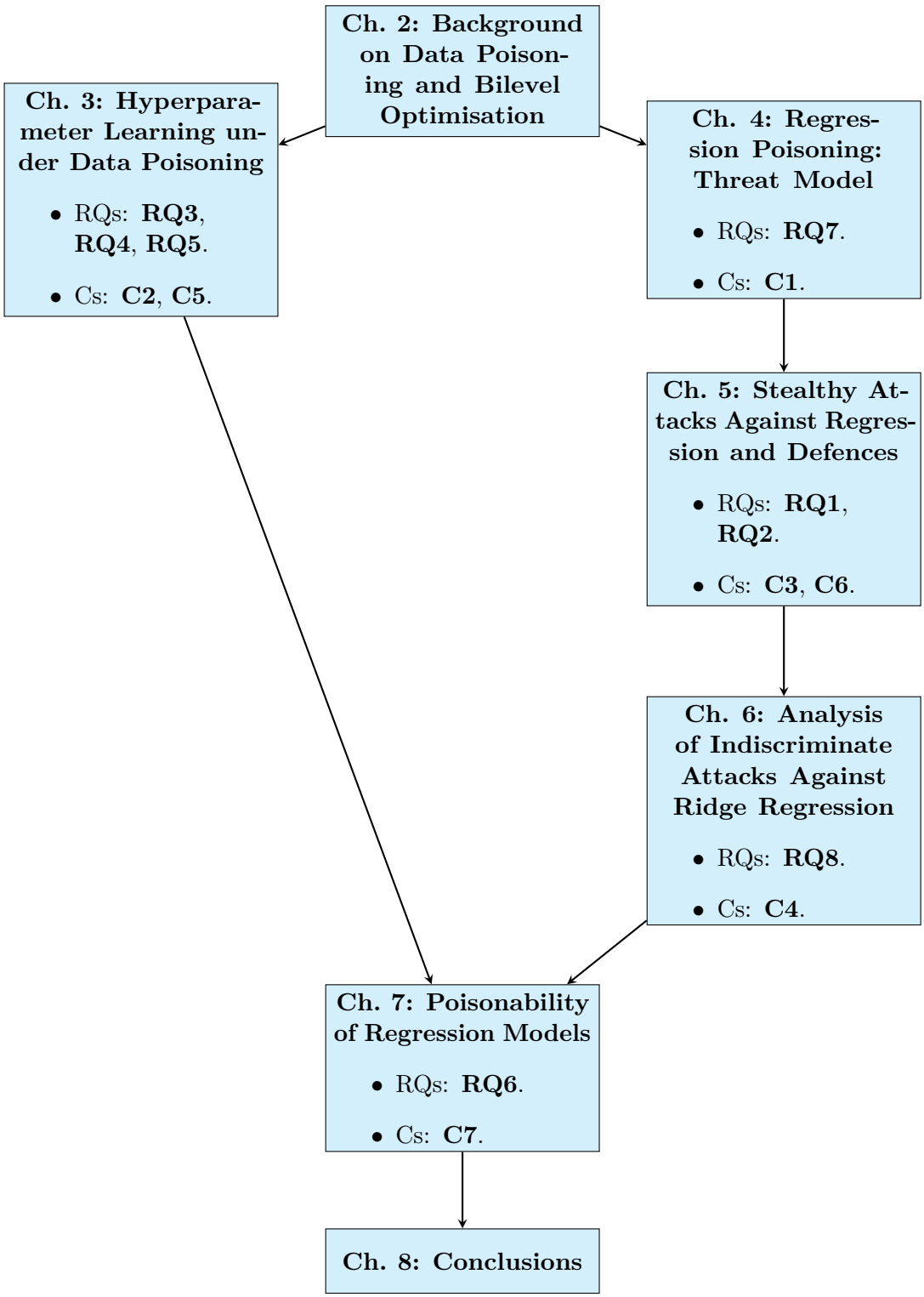


Figure 1.2: Thesis outline. Each arrow implies that the corresponding previous chapter is needed to understand the current chapter. We also include the research questions (RQs) and contributions (Cs) of each chapter.

a Bayesian linear regression to detect strong and stealthy attacks, which also improves on previous defences when the number of poisoning points is large.

Analysis of indiscriminate attacks against ridge regression: Ch. 6 theoretically characterises the attacker's objective function in the optimal indiscriminate attacks proposed in Ch. 5 . This includes the analysis of its convexity, asymptotic behaviour, critical points (stationary points), combinations of poisoning labels, equivalent formulations, influence of detectability constraints, and challenges in the optimisation process.

Poisonability of regression models: based on the limitations of current defences observed in Ch. 5, Ch. 7 introduces the concept of poisonability of a model. This measure aims to determine the number of poisoning points needed so that a certain tolerable mean error of the poisoning points on the poisoned model matches the mean error of the clean points on the poisoned model. Then, we analyse the poisonability of linear regression models.

Conclusions and future work: finally, the summary of achievements of this thesis, implications, and potential future work are described in Ch. 8.

Chapter 2

Background on Data Poisoning and Bilevel Optimisation

Synopsis

We review the state of the state art of adversarial ML and data poisoning attacks and defences, and describe the foundations of bilevel optimisation problems, which are the mathematical framework used to model optimal poisoning attacks. For a friendly overview of data poisoning attacks, also see:

- "Data Poisoning in AI" (2023). *Imperial College Podcast* [**Link**].
- "Defense Against the Dark Arts and Potions: AI Models Can Be Easily Poisoned" (2023), *T3chFest* [**Link**].

The particular state of the art specifically related to the claimed contributions is included in Ch. 3, 4, 5, and 7.

2.1 Machine Learning

In 1959, Samuel (Samuel, 1959) defined ML as the field of study that gives computers the ability to learn without being explicitly programmed. Some years afterwards, Mitchell (1997) expanded the definition as: "A computer program is said to learn from

experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". For instance, in an email program that watches which emails a particular user does or does not mark as spam, watching the user labelling emails as spam or not spam (ham) would be the experience E, the task T would be classifying emails as spam or ham, and the fraction of emails correctly classified as spam/ham would be the performance measure P.

There are several different types of learning algorithms, the main five types are called supervised learning, semi-supervised learning, unsupervised learning, self-supervised learning, and reinforcement learning (RL). Only the first shall be considered in this thesis.

2.1.1 Supervised Learning

Supervised learning techniques are methods that are given training examples in the form of inputs labelled with corresponding outputs. The term supervised learning originates from the view of the target being provided by an "instructor" or "teacher" who "shows" the ML system what to do (Goodfellow et al., 2014b). Generally, the goal is to estimate a mapping between vector-valued inputs, $\mathbf{x}$, and vector-valued outputs,[1] $\mathbf{y}$, during the *training phase*, such that in the inference (test) phase, the algorithm is able to predict the outputs for unseen inputs. If the output domain is categorical, the task is called classification, where the goal is to differentiate between two or more distinct classes of data. On the other hand, if the output is cardinal, the task is regression, where a value for a continuous magnitude that cannot be observed, is proposed given a set of features.

We can describe the goal of the learning algorithm as estimating a hypothesis function, $f(\mathbf{x})$, such that it predicts the corresponding output. For classification and regression tasks, a model (e.g., logistic regression, neural networks, linear regression, etc.) is characterised by a particular hypothesis function—or family of functions—that acts as a prior assumption over the data distribution. To obtain useful results for a finite number of samples, we must restrict the eligible solutions to a smaller set of functions. How to decide on the nature of the restrictions is based on considerations outside of the data. These restrictions are sometimes encoded via the parametric representation of $f_{\mathbf{w}}(\mathbf{x})$, or may be built into the learning method itself, either implicitly or explicitly. In general, the constraints imposed by most learning methods can be described as complexity restrictions of one kind or another.

[1] For the sake of simplicity, from now on we will consider a scalar-valued output y for every input $\mathbf{x}$.

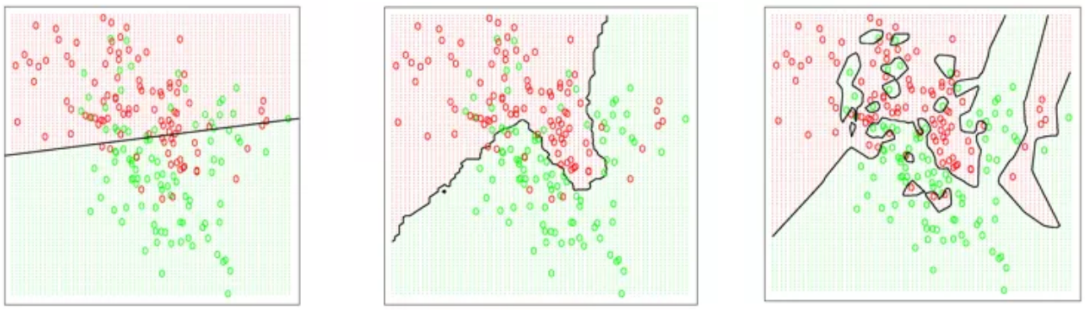
This usually means some kind of regular behaviour in small neighbourhoods of the input space. That is, for all input points $\mathbf{x}$ sufficiently close to each other in some metric, $f(\mathbf{x})$ exhibits some special structure such as nearly constant, linear or low-order polynomial behaviour. The estimator is then obtained by averaging or polynomial fitting in that neighbourhood (Hastie et al., 2009).

The nature of the constraint depends on the metric used. Some methods, such as kernel and local regression and tree-based methods, directly specify the metric and size of the neighbourhood. For instance, nearest-neighbour methods are based on the assumption that, locally, the function is constant: close to a target input $\mathbf{x}_0$, the function does not change much, and so close outputs can be averaged to produce $f(\mathbf{x}_0)$. Other methods such as basis-functions methods and neural networks implicitly define neighbourhoods of local behaviour. In particular, any method that attempts to produce locally varying functions in small isotropic neighbourhoods will run into problems in high dimensions—a.k.a. the *curse of dimensionality*. And conversely, all methods that overcome the dimensionality problems have an associated—and often implicit or adaptive—metric for measuring neighbourhoods, which basically does not allow the neighbourhood to be simultaneously small in all directions. (Hastie et al., 2009).

In the case of parametric methods, which will be focus of this thesis, the parameters $\mathbf{w}$ of these models are tuned to minimise a loss function[2] that measures the difference between the target values—ground truth—and the values predicted by the hypothesis function. This process is similar to human learning (Figueiras-Vidal, 2013), where we can learn the function from our perceptions (inputs) and our memory (parameters). In this case, the cost is related to personal assessments (including principles, emotions, and feelings) of the error.

Classic examples of classification tasks include: object recognition in images, voice classification, and spam filtering; and in the case of regression, predictive analytics, depth estimation in images, and imputation of missing values.

[2]In classification settings, the loss function is generally used as a proxy for the error rate in the optimisation problem, as the error is a step function (uses rounded predictions) and therefore not feasible for gradient-based optimisation.



(a) (b) (c)

Figure 2.1: Synthetic binary classification problem. The red and green points represent the instances for each class, and the solid black curves some possible decision boundary. **(a)** The curve is underfitting. **(b)** The plot describes the trend of the data reasonably well. **(c)** The decision boundary is overfitting (Troy, 2016).

2.2 Classical Assumptions for Machine Learning and Security Implications

An ideal model is an oracle that simply knows the true probability distribution that generates the data. Even such a model will still incur some error on many problems, because there may still be some noise in the distribution (Goodfellow et al., 2016). Unfortunately, in ML generally we do not know this probability distribution. Hence, assuming that the ML problem is solvable (i.e., the data we have access to is sufficient to explain the task), the central challenge is not only performing correctly on inputs that our model is trained on (typically called training data), but also on new, previously unseen inputs (test data). The ability to perform well on previously unobserved inputs is called generalisation (Goodfellow et al., 2016).

In this sense, we usually make a set of suppositions known as the IID assumptions. These assumptions are such that the examples in each dataset are independent from each other, and that the training and test sets are identically distributed, drawn from the same probability distribution as each other. The same distribution is then used to generate every training and test example (Goodfellow et al., 2016). This means that it is generally presupposed that the training and test data are representative of the whole data distribution, to which we do not have complete access.

The IID assumption is violated in adversarial environments, as the attacker specifically crafts malicious instances, given their influence, goals and knowledge of the system, as we will explain in Subsect. 2.3.2. These malicious instances are hence characterised

by probability distributions different, to a certain degree, from the one that generated the genuine instances. Furthermore, each one of the malicious points is not necessarily sampled from the same "malicious" data distribution. The data distribution is therefore non-stationary: it changes with the addition of the crafted instances. The independence assumption is broken as the presence of genuine samples affects directly or indirectly the probability of occurrence of malicious ones, as the latter are on purpose crafted according to information from true instances.

We can also reason about these implications by having a look at the classic dilemma in ML: underfitting and overfitting. Underfitting happens when the model is not able to obtain a sufficiently low error value on the training set, because the hypothesis function is too simple (it exhibits high bias, as in the left plot of Fig. 2.1). Overfitting occurs when the gap between the training error and test error is too large (Goodfellow et al., 2016). This takes place when the hypothesis function lies too close to the data points, usually as a result of high function complexity and large weights (i.e., high variance). Hence, it can appear uneven (right plot of Fig. 2.1) and does not generalise well on test data.

If an attacker inserts poisoning samples into the training dataset, these samples can subvert the learning algorithm and therefore change the value of its parameters. In this manner, the hypothesis function (which generates the decision boundary in classification problems and the regression curve in regression problems) will be altered and hence overfit to those samples, making the algorithm output wrong predictions on genuine and/or maliciously crafted test samples. If we think of overfitting from the defender's perspective, reducing the capacity of a model—e.g., through regularisation—can help make it more robust against poisoning samples, but the price may be to reduce its accuracy with respect to genuine samples. Then, the Occam's razor principle[3] (Blumer et al., 1987) for statistical learning theory no longer applies in adversarial settings, where the model simplicity is subject to the robustness to malicious data. This presents some fundamental implications on the trade-off between accuracy and robustness of ML algorithms, which will be a central topic of discussion in this thesis, particularly in Ch. 3.

[3]In statistical learning theory, the Occam's razor principle (invoked from the principle of parsimony) states that amongst different models with similar performance, one should choose the less complex one (Blumer et al., 1987).

2.3 Adversarial Machine Learning

Adversarial ML is a field of research that can be viewed in terms of the confrontation between a defender trying to learn securely, and an adversary trying to interfere. As the defender develops new strategies to learn securely, the attacker's strategy also evolves to overcome them in a continuous cycle (Huang et al., 2011). Thus, this field seeks to strengthen ML techniques against adversaries who wish to compromise the algorithm or obtain some information from the data and/or the model.

Pioneer studies on adversarial ML date back to 2004, when Dalvi et al. (2004), and later on Lowd and Meek (2005a,b) analysed the problem in the context of spam filtering, showing that linear classifiers could be easily tricked by few carefully-crafted changes in the content of spam emails. In 2006, Barreno et al. (2006) questioned the suitability of ML in adversarial settings from a broader viewpoint, taxonomising attacks and potential countermeasures both at training and at test time. Other notable works such as (Barreno et al., 2010; Huang et al., 2011) followed up these lines. Studying the security and privacy of ML has received considerable attention since 2013, when Szegedy et al. (2014) and subsequent studies (Goodfellow et al., 2014b; Nguyen et al., 2015; Moosavi-Dezfooli et al., 2016) showed that DNNs for image classification can be fooled by input images perturbed in an imperceptible manner (Biggio and Roli, 2018). Since then, and independently from the discovery of adversarial examples against deep networks (Szegedy et al., 2014), numerous papers have been published to reveal vulnerabilities of ML algorithms of all sorts including state-of-the-art ones such as deep convolutional networks, both at training time (poisoning/causative) and at test time (evasion/exploratory).

2.3.1 The Importance of Adversarial Machine Learning

The security and safety properties of a system that includes learning functionalities only make sense in the context of the entire system and depends, amongst other important aspects, on the security of the ML model itself. For instance, if the availability of the intrusion detection model embedded in a software-defined networking controller cannot be guaranteed, the availability of that controller and thus of the entire network it manages can be also affected (Papernot, 2018). In another example, the security of the passengers of self-driving vehicles (or the future self-flying planes) strongly relies on the security of

their sensors and learning systems. It is not difficult to imagine the dramatic consequences if such systems do not operate properly.

As Chio and Freeman (2018) remark, studying the security of ML is challenging as most ML solutions behave as black boxes, especially in the case of deep learning architectures. This means that the lack of interpretability and explainability of these models makes it difficult for developers and users to understand the rationale of their predictions in terms of the logic of the system. Then, in many cases it is not easy to detect when a learning system has been compromised by an adversary. In consequence, it is crucial to assure the stability and robustness of ML systems—by understanding systematic vulnerabilities and developing new testing methodologies—to authorise their safe deployment and acceptance in security-sensitive applications. In addition, this leads to the need for auditing these systems to verify they only solve the task for which they were designed—i.e., do not present backdoors (a.k.a. Trojans (Gu et al., 2019)).

On the other hand, it is noteworthy that the potential vulnerabilities of learning-based systems are not limited to the ML model itself but also encompass the whole data processing pipeline: data collection from individuals, sensors and devices in the physical domain; preprocessing (including data annotation, cleaning, and dimensionality reduction) in the digital representation; training (ML model itself) and model interpretation (including explainability, auditability and fairness); and deployment (inference phase) and accountability, again in the physical domain. Obviously, the security of the ML model also relies upon the security of the computing system that hosts it—which is out of the scope of this thesis. For instance, the privacy of the data analysed by an ML algorithm could be endangered by a side-channel attack on the hardware accelerator on which the model is trained and/or deployed (Papernot, 2018); or in the case of outsourced learning (Li et al., 2018; Gu et al., 2019), transfer learning (Pan and Yang, 2010; Gu et al., 2019) or federated learning (Konečnỳ et al., 2016; Bagdasaryan et al., 2020), the security and privacy of the model and data could be compromised if the telecommunications network were jeopardised.

2.3.2 Threat Model

Like other traditional security contexts, the first step towards the analysis of the vulnerabilities of ML systems requires to define an appropriate threat model (Muñoz-González

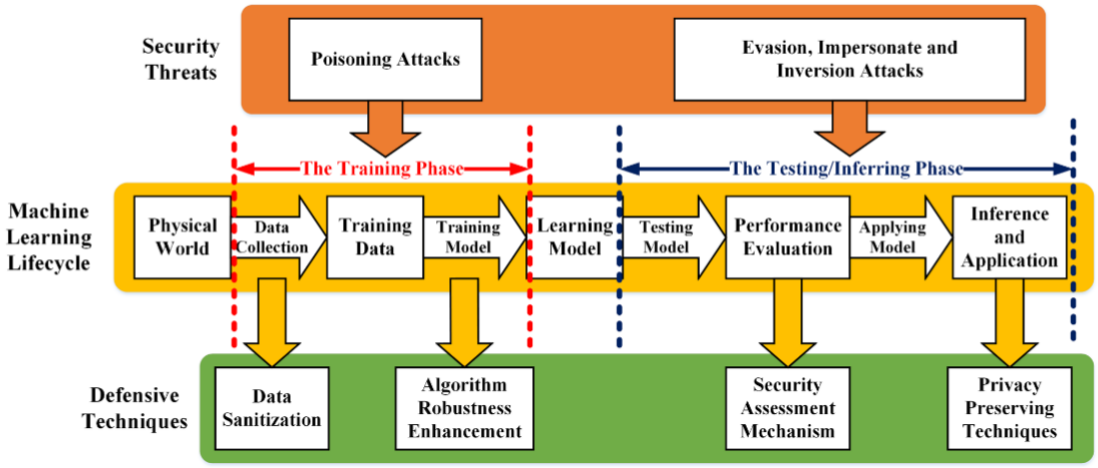


Figure 2.2: Illustration of security threats and defensive techniques of ML (Liu et al., 2018a).

and Lupu, 2019). This model, synthesised from (Barreno et al., 2006, 2010; Huang et al., 2011; Biggio et al., 2014; Muñoz-González et al., 2017; Muñoz-González and Lupu, 2019) in the next subsection, characterises the security of learning systems as a function of the adversarial capabilities, goals and knowledge. A summary of the security threats and defensive techniques considering the ML pipeline (Liu et al., 2018a) is shown in Fig. 2.2.

Attacker's Capability

This characteristic depends on the influence that the attacker has on the input data, and on application-specific data manipulation constraints (Muñoz-González and Lupu, 2019).

- **Attack Influence**: depending on whether the adversary can manipulate training and/or test data, we can differentiate three main type of attacks:
  - **Poisoning/causative attacks**: the attacker can manipulate the training data—which is untrusted—to subvert the learning algorithm (Muñoz-González and Lupu, 2019), destabilising the system. Poisoning attacks can also facilitate future evasion attacks, in which the adversary can influence both training and test sets, a.k.a. backdoor attacks.
  - **Backdoor/Trojan attacks** (Gu et al., 2019; Liu et al., 2018b; Chen et al., 2017b): the attacker injects a trigger—e.g., in the case of image classification, a common sticker—into a set of training examples, which causes the model to misclassify samples with the same trigger pattern at test time, without affecting

the performance of genuine samples. Backdoor attacks can also be interpreted as a particular case of targeted poisoning attacks.

- **Evasion/exploratory attacks**: the attacker can only manipulate data at test time. They can probe the system to learn the weaknesses and blind spots to produce intentional errors (depending on its robustness); or to obtain information about the ML model, the data employed for training, and/or the users (privacy and confidentiality violation) (Muñoz-González and Lupu, 2019).

- **Data manipulation constraints**: the data must always have valid attribute values. Moreover, depending upon the application domain, the attacker may be limited to manipulate the features or the labels of the attack samples. The attacker may also consider additional constraints to avoid detection (detectability constraints), e.g., from outlier detectors (Paudice et al., 2018a). These constraints can be generally included in the definition of the optimal attack strategy, by assuming that the initial poisoning samples $\mathcal{D}_{\mathrm{p}}$ can only be modified according to a space of possible modifications $\Phi(\mathcal{D}_{\mathrm{p}})$, as for example by bounding the norm of the input perturbation of each attack point (Muñoz-González and Lupu, 2019; Biggio and Roli, 2018).

Attacker's Goal

This aspect is defined in terms of the desired security violation, attack specificity, and error specificity (the latter defined in classification settings), as detailed below (Muñoz-González and Lupu, 2019).

- **Security violation**: this characteristic defines the high-level security violation caused by the attack (Muñoz-González and Lupu, 2019). We can distinguish three different security violations against ML systems:

  - **Availability violation**: when the attacker aims to compromise system functionality, e.g., by increasing the overall error rate in a classification algorithm. If a particular system depends on the performance of the ML model to make decisions that impact its availability, these attacks fall under the broad category of denial of dervice (Papernot et al., 2018).
  - **Integrity violation**: when the attack evades detection without compromising normal operation of the system. Attacks attempting to control concrete model

outputs are at the heart of integrity attacks (Papernot et al., 2018). For example, producing false negatives in binary classification problems.

- **Privacy** and **confidentiality violation**: if the attacker obtains sensitive information about the ML system, the data used for training, or the users of the system. ML models have enough capacity to capture and memorise elements of their training data (Fredrikson et al., 2015). As such, it is hard to provide guarantees that participation in a dataset does not harm the privacy of an individual (Papernot et al., 2018). This form of security violation will not be taken into consideration in our research, but is relevant for privacy-preserving learning applications (Huang et al., 2011).

- **Samples specificity**: this axis defines how specific the adversary's intention is. This is not a clear categorisation into two possible specificity values, but represents a continuous spectrum (Muñoz-González and Lupu, 2019):
  - **Indiscriminate attacks**: when the attacker aims to degrade the performance of the system or to produce errors in an indiscriminate manner, i.e., for a broad set of samples.
  - **Targeted attacks**: when the attacker aims to produce errors or to degrade the system performance for a reduced set of samples.
  - **Subpopulation attacks**: when the attacker aims to impact a subpopulation,[4] while retaining model accuracy on the rest of the test samples (Jagielski et al., 2021). Subpopulation attacks interpolate between targeted and indiscriminate attacks (depending on the subpopulation size).

- **Error specificity**: this characteristic was introduced by Muñoz-González et al. (2017) to disambiguate cases where the nature of the errors can be different, as in multiclass classification tasks (Muñoz-González and Lupu, 2019). Hence, according to the error specificity, an attack can be defined as:
  - **Confidence reduction attacks**: they reduce the confidence of the model's predictions (introducing class ambiguity) (Papernot et al., 2016b). This does not imply that they explicitly increase some kind of error, as the error is a step function (but it could facilitate other attacks).

[4] Subpopulations consist of points with similar feature representations, and the size of the subpopulations determines the overall impact of these attacks.

- **Error-generic attacks**: when the attacker aims to produce errors in the system, regardless of the kind of error.
- **Error-specific attacks**: when the attacker aims to produce specific types of error in the system.

Attacker's Knowledge

The attacker can have different levels of knowledge about the targeted ML system including (Muñoz-González and Lupu, 2019):

- Training dataset, $\mathcal{D}_{\text{tr}}$. This includes the features of the samples and their corresponding labels.
- Feature representation used by the learning algorithm, $\mathcal{X}$. This refers to the actual features used by the algorithm after the data preprocessing stage. This data preprocessing can also include dimensionality reduction techniques.
- Learning model, $\mathcal{M}$. The attacker can know the learner's inductive bias,[5] e.g., hypotheses such as linearity, maximal margin, polynomial, basis functions, etc.; and the architecture of the model.
- Hyperparameters of the model (priors), Λ.
- Parameters of the model (if the model is parametric), $\mathbf{w}$.
- Loss function that the learning algorithm aims to optimise, $\mathcal{L}$.

The attacker's knowledge can be therefore characterised in terms of a space Θ, whose elements encode the aforementioned components as $\Theta = (\mathcal{D}_{\text{tr}}, \mathcal{X}, \mathcal{M}, \Lambda, \mathcal{L}, \mathbf{w})$. This representation includes many different attack scenarios, depending on the assumptions made on each of the components of Θ. But typically, three main settings are considered (Muñoz-González and Lupu, 2019):

- **Perfect-knowledge (PK)**: the attacker is assumed to know everything about the targeted system. Therefore, $\Theta_{\text{PK}} = (\mathcal{D}_{\text{tr}}, \mathcal{X}, \mathcal{M}, \Lambda, \mathcal{L}, \mathbf{w})$. Although unrealistic in

[5]The inductive bias of a learning algorithm is the set of assumptions that the learner uses to predict outputs of previously unseen inputs. A classic example of an inductive bias is Occam's razor, assuming that the simplest consistent hypothesis about the prediction function is actually the best.

most cases, perfect knowledge attacks allow to perform worst-case evaluations of the security of the ML system. This enables an empirical estimation—which can be performed with gradient-based optimisation—of upper bounds on the performance degradation of the system when is under attack (Muñoz-González and Lupu, 2019). This will be the case in this thesis. It can also be helpful for model selection, comparing the performance of different learning algorithms with different settings when they are under attack (Muñoz-González and Lupu, 2019). PK settings are also appealing to establish an analysis of the trade-off between the accuracy and robustness of the system, which we will study in Ch. 3.

- **Limited-knowledge (LK)**: in most research literature, limited knowledge attacks typically consider two cases: attacks with surrogate data and attacks with surrogate learner (Muñoz-González and Lupu, 2019):
  - **Surrogate data (SD)**: the attacker knows the feature representation, the learning algorithm, and the loss function. These attacks also assume that the attacker has access to a surrogate dataset, $\hat{\mathcal{D}}_{\text{tr}}$, with similar characteristics and data distribution than $\mathcal{D}_{\text{tr}}$. Then, the attacker estimates the parameters $\hat{\mathbf{w}}$[6] of the learning algorithm $\hat{\mathcal{D}}_{\text{tr}}$ by optimising the loss function on the surrogate dataset $\hat{\mathcal{D}}_{\text{tr}}$ (Muñoz-González and Lupu, 2019). Hence, $\Theta_{\text{LK}-\text{SD}} = (\hat{\mathcal{D}}_{\text{tr}}, \mathcal{X}, \mathcal{M}, \Lambda, \mathcal{L}, \hat{\mathbf{w}})$.
  - **Surrogate learner (SL)**: the attacker is assumed to know the training data and the feature representation (e.g., if the learning algorithm is trained on publicly available datasets (Muñoz-González and Lupu, 2019)), but not the learning algorithm and the loss function (Muñoz-González and Lupu, 2019). Therefore, $\Theta_{\text{LK}-\text{SL}} = (\mathcal{D}_{\text{tr}}, \mathcal{X}, \hat{\mathcal{M}}, \hat{\Lambda}, \hat{\mathcal{L}}, \hat{\mathbf{w}})$. In this case, the estimated parameters $\hat{\mathbf{w}}$ may also belong to a different vector space than the ones of the targeted system, as the model of the attacker can be different (Muñoz-González and Lupu, 2019). These attacks also include the cases where the corresponding optimisation problem is intractable or too difficult to solve, even if the attacker knows the learning algorithm of the targeted system. Then, a surrogate model can help avoid this difficulty by crafting a tractable but (possibly) less effective attack (Muñoz-González and Lupu, 2019). Moreover, surrogate-learner environments are commonly used as a procedure to evaluate the transferability of attacks

[6]The hat symbol denotes limited knowledge of a given component.

between learning algorithms (Biggio and Roli, 2018): the attacker can craft attacks against a surrogate model and test them against the targeted one (Muñoz-González et al., 2017).

- **Zero-knowledge (ZK)**: the attacker has no substantial knowledge of the training set, the learning model and its parameters and hyperparameters, or the loss function. However, they know that the learning system is designed to perform some particular task, and have an idea of which potential transformations to apply to cause some feature changes. This means that knowledge of the feature representation may be partial (Biggio and Roli, 2018). Thus, $\Theta_{\text{ZK}} = (\hat{\mathcal{D}}_{\text{tr}}, \hat{\mathcal{X}}, \hat{\mathcal{M}}, \hat{\Lambda}, \hat{\mathcal{L}}, \hat{\mathbf{w}})$. As crafting poisoning samples typically requires some degree of knowledge about the model, most ZK attacks have been proposed for evasion scenarios (Biggio et al., 2013; Papernot et al., 2016a, 2017; Chen et al., 2017a; Ilyas et al., 2019; Co et al., 2019).
  - **Oracle**: without access to the training data or the ML model, the adversary has the ability to query the targeted model to obtain output classifications from carefully crafted inputs. This enables them to observe and extract information (or even reconstruct the model) from the relationship between changes in inputs and outputs, to adaptively craft adversarial examples (derivative-free optimisation (Snoek et al., 2012; Shahriari et al., 2016)). Thus, a key metric when comparing different attacks is the wealth of information returned by the oracle, and the number of oracle queries they perform (Papernot et al., 2016b, 2017). This is also a common methodology to characterise the transferability of attacks between models.
  - **Samples**: the adversary has the ability to collect pairs of input and output related to the learning model. However, they cannot modify these inputs to observe the difference in the output. These pairs are only useful in very large quantities (Papernot et al., 2018).

In the previous analysis we have not considered the case in which the learner is applying some particular defence strategy. Therefore, this component can be added to the space of possibilities for the attacker's knowledge. The attacker can thus target this defence by means of imposing detectability (data manipulation) constraints in the formulation of the optimisation problem, and use adaptive attacks or find novel procedures to bypass the proposed defence. As a result, the problem of adversarial and poisoning examples is very

challenging given enough computation, time, and insider knowledge of the targeted model. However, some techniques can help mitigate the effect of this problem and enhance the security of ML systems against this threat.

Regarding the knowledge of the target defence, there are three different scenarios: (ii) the adversary is aware of the defence strategy and targets it as well; (ii) they have the knowledge that a certain defence is deployed, but actually do not know which concrete strategy, so they target a reasonable surrogate defence (e.g., strong defences publicly accessible); and (iii) the adversary does not take into consideration any defence. In the second case, they can also establish a trade-off for the attack impact in terms of the surrogate defence and the case in which there is no defence. We will develop this strategy in Ch. 5. Also, it could happen that the defender is deploying a honeypot—for instance, fake adversarial regions (Shan et al., 2020)—to gain information about the attacker, so that the latter should be cautious when crafting the malicious samples.

Next, we review the main kind of attacks we shall tackle in this thesis, i.e., data poisoning attacks.

2.4 Data Poisoning

In many applications, ML systems rely on data collected from *untrusted* data sources, such as humans—e.g., product reviews, user feedback on social media, or insurance claims—machines, sensors, or Internet-of-Things (IoT) devices that can be compromised and manipulated. Malicious data from these compromised sources can then be used to poison the learning algorithms themselves. In other applications, the labelling of the training datasets is done manually, and *crowdsourcing* techniques are used to aggregate the labelling information from human annotators. In these cases, *crowdturfing* attacks are possible where malicious annotators collude to deceive the crowdsourcing algorithm by manipulating some of the labels (Yao et al., 2017).

Other scenarios include federated learning (McMahan et al., 2017), which enables thousands of participants to construct a learning model without sharing their private training data with each other, but where any participant could introduce malicious functionality into the joint global model (Bagdasaryan et al., 2020; Fung et al., 2018).

These situations expose ML algorithms to data poisoning attacks—a.k.a. causative

attacks—where adversaries manipulate a fraction of the training data to subvert the learning process, either to decrease its overall performance or to produce a particular kind of error in the system (Barreno et al., 2010; Huang et al., 2011; Muñoz-González et al., 2019a). Data poisoning attacks can also facilitate subsequent evasion attacks or produce *backdoor* (or *Trojan*) attacks (Gu et al., 2019; Liu et al., 2018b). The adaptability of ML algorithms—when inferring patterns from malicious samples—makes data poisoning attacks one of the most relevant and emerging threats for data-driven technologies. For instance, according to a survey carried out by Microsoft in 2020 (Kumar et al., 2020), poisoning is the top perceived threat against ML for business decision makers.

The first poisoning attacks reported in the literature targeted specific applications, such as spam filtering (Nelson et al., 2008; Barreno et al., 2010) or anomaly detection (Barreno et al., 2006; Kloft and Laskov, 2012). In particular, classic poisoning attacks appeared in the context of spam filtering with ML systems such as SpamBayes (naive Bayes for spam classification). SpamBayes retrains itself periodically with the new data that it has collected. This allows attackers to place their malicious data (emails) into the training dataset of the SpamBayes filter. By sending malicious emails into the user's inbox, the latter labels it as spam. Later, when the learning algorithm retrains itself, the malicious emails can become part of the training dataset. For example, in dictionary attacks, the adversary constructs malicious instances by inserting words sampled from a distribution commonly found in regular emails—dictionary—into the messages. This is done with the intent of associating all words with spam (Lowd and Meek, 2005b). When a spam filter is trained on this sort of messages, it starts to associate those words with spam email, raising its overall false positive rate. In contrast, in ham attacks, words that are characteristic of the ham (benign) class are included in crafted emails, making it a more efficient version of the dictionary attack.

A well-known real-world poisoning attack is the case of Tay, a chatbot that was released by Microsoft on Twitter in 2016. The chatbot caused controversy when it began to post offensive tweets through its Twitter account, forcing Microsoft to shut down the service only 16 hours after its launch (Neff and Nagy, 2016). According to Microsoft, this was caused by "trolls" who "attacked" the service as the bot made replies based on its interactions with people on Twitter. In a certain way, Twitter users "poisoned" Tay's data with their deliberately offensive tweets, which influenced Tay's own tweets.

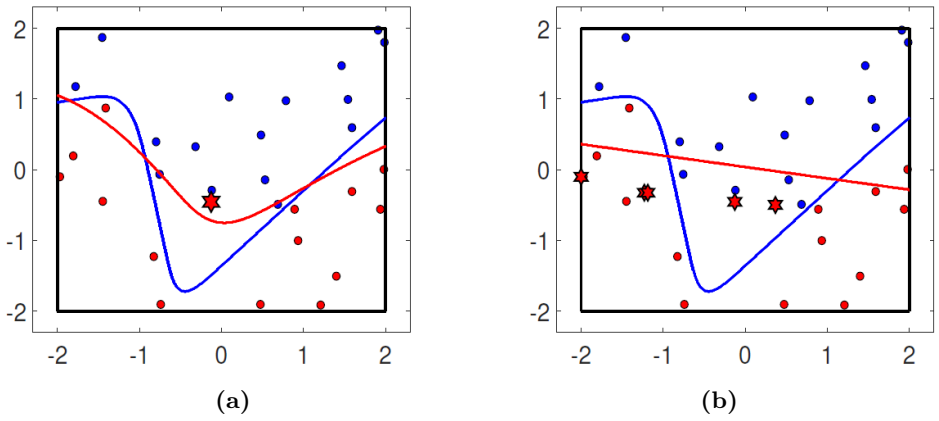


Figure 2.3: Synthetic example of data poisoning in a binary classification problem. The points denote training examples, and the colours denote distinct classes. Starred points indicate poisoning points intentionally positioned to distort the decision boundary. The red line denotes the learnt decision boundary and the blue line, the true decision boundary. **(a)** Effect on the decision boundary after injecting one poisoning point. **(b)** Effect after injecting five poisoning points (Muñoz-González and Lupu, 2019).

More generally, in data poisoning attacks the adversary has influence over a fraction of the training data used by the learning algorithm—by injecting or substituting malicious data points. The attack strength is normally measured in terms of the relative number of poisoning samples with respect to the size of the training data. The large size of the datasets usually means it is not possible to filter out the manipulated data (Paudice et al., 2018a). The attacker's goal is therefore to destabilise the system by subverting the learning process—modifying the decision boundary or regression curve—in an indiscriminate or targeted way, either to decrease the overall performance of the system or to produce a particular kind of error (Muñoz-González and Lupu, 2019). These attacks can also facilitate subsequent evasion of the system, i.e., backdoor or Trojan attacks (Gu et al., 2019; Liu et al., 2018b).

In Fig. 2.3 we can observe an example of a poisoning attack against a small neural network classifier, where the system aims to correctly classify the blue and the red dots. We can appreciate that the decision boundary changes (from the blue line to the red one) after adding several malicious points (red stars), which cause the system to increase its error rate in the test phase (Muñoz-González and Lupu, 2019).

In the last years, in-depth research on poisoning attacks has brought a more comprehensive taxonomy that generalises them to other application domains and learning algorithms. For the attacker to craft successful poisoning samples, the general assumption is that the attacker has some degree of knowledge about the training data, and possibly

the learning model or a surrogate model with similar representational power (e.g., in terms of similar decision boundary). In this sense, modifications of the training data can be seen as altering the distribution that generated the training data, thus creating a mismatch between the distributions used for training and test phase (Barreno et al., 2010; Dalvi et al., 2004).

Next, we highlight the main sort of poisoning attacks we can find in the research literature.

2.4.1 Data Poisoning Attacks Algorithms

Different poisoning attack strategies have demonstrated the sensitivity of ML algorithms to this threat, including support vector machines (SVMs) (Biggio et al., 2012; Steinhardt et al., 2017; Koh et al., 2022), logistic regression (Muñoz-González et al., 2017; Koh et al., 2022), or neural networks and deep learning architectures (Muñoz-González et al., 2017; Koh and Liang, 2017), amongst others. Here we summarise the most popular data poisoning attacks.

- **Label-flipping attacks**: this is simplest strategy to implement poisoning attacks; the attacker does not perturb the feature values in the training data, but merely changes the labels of a fraction of the training data so that they are incorrect (Biggio et al., 2011b; Paudice et al., 2018b). The method by which these labels are changed is also a matter of strategy; one could simply flip labels randomly, or one could utilise a more precise method, which increases the loss of the defender optimally depending on the training sample that is flipped.

- **Implicit differentiation and influence functions**: early works on data poisoning against classic models, e.g., SVMs, logistic regression, and linear regression, require that the inner problem is solved exactly—by leveraging its stationarity conditions—and then using the implicit function theorem to compute the gradients w.r.t. the poisoning points, which is computationally very inefficient (Biggio et al., 2012; Xiao et al., 2015; Mei and Zhu, 2015; Jagielski et al., 2018). Koh and Liang (2017) created adversarial training examples by exploiting influence functions. This approach allows to craft successful targeted attacks by injecting small perturbations to genuine data points in the training set.

- **Iterative differentiation**: these attacks compute the gradient for the poisoning points by approximately solving the inner problem using a reduced number of steps of gradient descent (Muñoz-González et al., 2017). A gradient ascent step is then conducted on the outer loss by back-propagating through the inner minimisation dynamics. This reduces the computational cost compared to implicit differentiation methods, but the method has still a limited impact against large deep neural network (DNN) architectures.

- **Generative models**: several methods propose to train generative models such as autoencoders (Yang et al., 2017; Feng et al., 2019), and generative adversarial networks (GANs) (Muñoz-González et al., 2019b) to produce poisoning samples against deep networks at scale. After the generative model is trained, poisoning samples can be crafted simply by conducting a forward pass through the generator or the autoencoder. Further, Muñoz-González et al. (2019b) proposed a mechanism to generate poisoning samples at different degrees of detectability that can surpass state-of-the-art defences. While this enables poisoning additional samples at little additional computational cost, the cost of training the original generative model is still high.

- **StatP**: Jagielski et al. (2018) propose a statistical attack that samples points from a multivariate Gaussian distribution, where the corresponding mean and covariance matrix are estimated from the clean training data. Then, StatP rounds the feature variables to the corners, queries the model and rounds the target variable to the opposite corner—exploiting the observation that the most effective poisoning points are near corners. The corners are defined as the minimum and maximum of the feasibility domain of each variable. In summary, this attack creates a few isolated clusters of adversarial data, where both features and target take only extreme values. However, this attack is only shown to be successful against linear models, and can be detected by defences that reject extreme values.

- **BetaPoisoning**: instead of solving a bilevel optimisation problem, Cinà et al. (2021) propose a suboptimal heuristic strategy, together with a variable reduction technique, called BetaPoisoning: crafting the poisoning points based on maximising a surrogate loss—approximated by a Gaussian kernel density estimator. They assume that the data distribution for each class is approximately "Gaussian" and linearly

separable. Then, they craft the poisoning points via a linear combination of validation points that maximises the surrogate loss. This makes the dataset no longer linearly separable, thereby hurting the accuracy of linear models. Their formulation is related to the evasion attack proposed by Biggio et al. (2013). A desirable property of the BetaPoisoning attack is that it does not need to retrain the model while crafting the poisoning samples.

Clean-Label Data Poisoning Attacks

We briefly detail prominent clean-label data poisoning attacks against DNNs. These attacks involve correctly labelled training data, and do not degrade the performance on non-targeted examples. This can also help make the attack less detectable, with the goal of controlling the behaviour of the classifier on specific test instances.

- **Feature Collision**: Shafahi et al. (2018) present a targeted poisoning attack that perturbs training data so that its deep features collide with those corresponding to a target image. At the same time, the attack penalises the size of perturbations applied to training data to maintain visual similarity between the original and perturbed images. In the attack developed by Aghakhani et al. (2021), the target image resides close to the centre of the convex hull of the poisoning images in feature space, to increase the success of feature collision. This method improves on the work of Zhu et al. (2019), which constructs a convex polytope around the target based on an ensemble of models. All these methods are efficient, but designed to attack fine-tuning scenarios where the feature extractor is nearly fixed and not influenced by poisoned data.

- **MetaPoison**: Huang et al. (2020) generate poisoning points based on unrolling the bilevel objective encountered in targeted data poisoning for several steps and optimising the unrolled objective over an ensemble of models at different stages in training, leading to an attack that is robust to new initialisations in from-scratch training as well as to changes in model architectures. This is the first attack to succeed against deep networks trained from scratch on CIFAR-10. However, unrolled ensembles remain costly. Furthermore, MetaPoison performs modestly on indiscriminate attacks. This could be attributed to the fact that the adversarial loss

for indiscriminate attacks, which takes into account all images in the validation set, places quite a few constraints on the poisoning points.

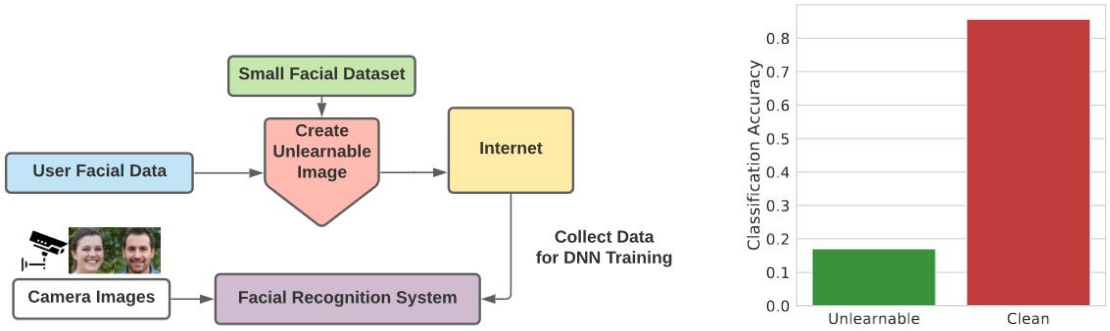
- **Gradient Matching**: Witches' Brew (Geiping et al., 2021a) instead approximates the bilevel objective by gradient matching of poisoning and target data, leading to a computationally efficient attack (in comparison to MetaPoison (Huang et al., 2020)), combining the efficiency of feature collision attacks with the success of bilevel approximations (i.e., from-scratch settings). It uses tools such as random restarts, differentiable data augmentation (including translation, crop, and horizontal flip), and model ensembles. It does not require updates of the model's parameters after each poisoning optimisation step. The bilevel approximation of Geiping et al. (2021a) can also adapted to indiscriminate attacks for neural networks by substituting the original targeted objective with an indiscriminate one (Fowl et al., 2021a).

Clean-Label Data Poisoning Attacks to Prevent Data Exploitation

Since DNNs usually require large datasets to train, many practitioners scrape data from external sources such as the Internet. However, an external data owner may not be willing to let this happen. This has raised public concerns about the "free" exploration of personal data for unauthorised or even illegal purposes (Huang et al., 2021). On this subject, an attacker can aim to slightly modify training data to spoil the training process such that a trained network lacks generalisation. For instance, this setting is relevant to practitioners like social media companies who wish to modify all user data to prevent others from using data scraped from their platforms. However, these attacks are effective when the attacker manipulates the whole training set—or a large fraction of it—which can be impractical in some situations.

- **TensorClog**: TensorClog (Shen et al., 2019) is an indiscriminate attack that perturbs training data so that neural networks have gradients of very low magnitude and thus do not train effectively—decreasing validation performance—while also ensuring that perturbed images are similar to the originals. However, this strategy is only effective in PK transfer learning settings when the feature extractor is known and fixed, and even in this idealised setting, the attack is only mildly successful (Shen et al., 2019).

- **Unlearnable examples**: Huang et al. (2021) intentionally optimise the perturbations



(a) (b)

Figure 2.4: Preventing exploitation of face data using error-minimising noise. In this scenario, the defender wants to prevent their face images from being exploited to train face recognition or verification systems. The user has access to a small dataset of facial images and uses this small dataset to generate and apply the error-minimising noise to their own face images before sharing them on online social media platforms. After this, the unlearnable version of face images on social media is collected to train a DNN-based facial recognition or verification system. The system is then used to recognise one of the user's clean face images captured somewhere by a camera. The goal is to prevent the DNN from learning the defender's face images and make it perform poorly on their clean face images (Huang et al., 2021).

to reduce the training loss and prevent data from being freely exploited by deep learning (see Fig. 2.4). The main motivation is that if the training loss is zero, then the target model will have nothing to learn as there is nothing to backpropagate. The inner optimisation aims to find the perturbations by minimising the model's classification loss, while the outer optimisation updates the model's parameters by training it on the perturbed images achieved in the inner optimisation. The above inner and outer optimisation share the same objective, and the model training starts from a random initialisation in each round while the perturbations accumulate through the whole optimisation. The inner and outer optimisation are alternatively implemented and finally terminated when the predefined training accuracy on the outer model is met. However, such conferred unlearnability is found fragile to adversarial training. Hence, Fu et al. (2022) improve upon the work of Huang et al. (2021) to generate robust unlearnable examples that are protected from adversarial training.

- **Poisoning points as adversarial examples**: Fowl et al. (2021b) show that the objective of adversarial examples, which is a single objective of maximising the loss, is sufficient to generate powerful data poisoning perturbations. They use the classic objective of adversarial examples. Surprisingly, Fowl et al. (2021b) demonstrate that

this simple objective can generate perturbations that achieve state-of-the-art attack performance.

- **Neural tangent generalisation attack**: Yuan and Wu (2021) adopt a Gaussian process based on neural tangent kernels as a surrogate model for the mean of a class of wide DNNs. This method can enhance the transferability of the generated noise across different models.

2.4.2 Optimal Poisoning Attacks

An important step towards building empirical assurance methodologies for learning systems is the formulation of optimal poisoning strategies. The purpose of these procedures is to determine the maximum damage an adversary can perpetrate if they have perfect knowledge about the system. This methodology is of great interest and is one of the core aspects of this thesis as it allows to derive worst-case analyses for different levels of data poisoning, as well as to develop effective mitigation strategies.

The core of optimal poisoning attacks is to solve a bilevel optimisation problem (which we shall review in Sect. 2.5), i.e., an optimisation problem that *depends* on another optimisation problem. In these cases, the attacker typically aims to maximise a malicious objective (e.g., to maximise the error for a set of target points) by manipulating a fraction of the training data. At the same time, the defender aims to optimise a different objective function to learn the optimal model's parameters, typically by minimising some loss function evaluated on the poisoned training set (Muñoz-González et al., 2017). Every time an attacker updates the value of a poisoning sample, they need to train the learning algorithm for a number of iterations. Then, computing optimal attacks is complex, and there is no general method suited to all classes of learning algorithms. Thus, extending this formulation to deep learning architectures is not a trivial task, as the large number of parameters limits its effectiveness. For instance, given that the loss function of deep learning algorithms is generally non-convex, the difficulty lies in reaching a high loss value that guarantees a significant increase in the test error rate.

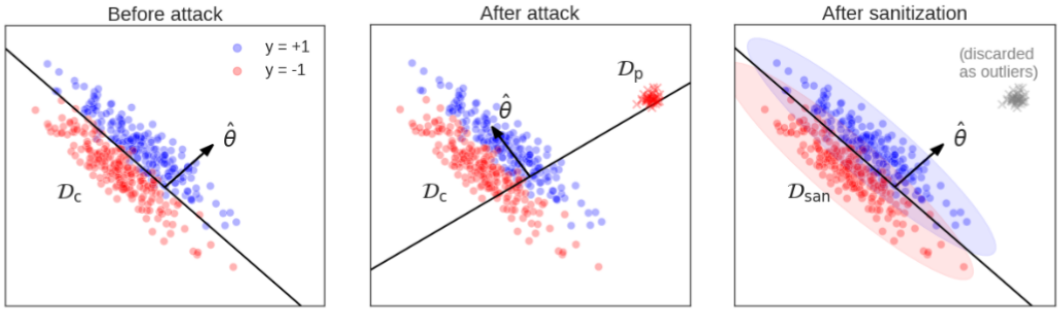
Prior work on optimal attack strategies focussed on computing optimal attack points against SVMs (Biggio et al., 2012; Mei and Zhu, 2015; Koh et al., 2022), logistic regression (Mei and Zhu, 2015; Koh et al., 2022), and embedded feature selection methods

(Xiao et al., 2015), via a computationally intensive gradient ascent strategy. These methods can effectively degrade the accuracy of the target model. However, the data generation is very time-consuming because of the expensive gradient calculation of the validation loss function w.r.t. the poisoning point. This computation involves implicit differentiation—the model's parameters are themselves a complex implicit function of the poisoning point—taking into account the inverse of the Hessian matrix (Biggio et al., 2012; Mei and Zhu, 2015; Koh et al., 2022) of the loss function. Hence, the computational cost of the poisoning data generation using implicit differentiation is proportional to the dimension of the input data and the complexity of the target model. Thus, the scalability of implicit differentiation methods may be a severe issue, especially considering the rapid growth of the sizes of state-of-the-art deep networks and their training datasets.

A more general approach was introduced by Muñoz-González et al. (2017), formulating different optimal attack strategies for targeting multiclass classifiers. The authors also proposed an algorithm to estimate the hypergradients in the corresponding bilevel optimisation problem through RMD (Domke, 2012; Maclaurin et al., 2015; Franceschi et al., 2017, 2018), which significantly improves the scalability of optimal attacks, allowing to poison a broader range of learning algorithms, including neural networks.

Koh et al. (2022) proposed an algorithm for solving bilevel problems with detectability constraints, allowing to craft poisoning points that can bypass outlier detectors. However, since the attacker's feasible set depends on the output of a predefined outlier detector, the algorithm is computationally demanding—which limits its applicability in practical scenarios. For instance, the attacker's feasible set depend on the model's parameters that the defender learns, which in turn depend on the poisoning points. This dependence makes it difficult for the attacker to explicitly constrain their poisoning points to lie within such feasible sets.

The suboptimal attacks proposed by Steinhardt et al. (2017); Koh et al. (2022) aim to simplify the bilevel optimisation problem by replacing the validation loss at the outer level with the training loss. This substitution changes the bilevel problem into a saddle-point problem—i.e., one that can be expressed in the form minimax—that can be solved efficiently via gradient-based optimisation. The core assumptions are that the test and training set (the latter including the poisoning samples) are drawn from the same distribution, that the training set is sufficiently representative, and the model is appropriately regularised, such



(a) **(b)** **(c)**

Figure 2.5: Data poisoning and anomaly detection. **(a)** In non-adversarial settings, the defender can often learn the model's parameters that (approximately) fit the clean training data. **(b)** The addition of poisoning data can significantly change the learnt parameters, leading to high test error. **(c)** By discarding outliers and then training on the remaining data, the defender can mitigate the effectiveness of the attacker. In this example, the defender discards all blue points outside the blue ellipse, and all red points outside the red ellipse (Koh et al., 2022).

that the validation loss is similar to the training loss—which is not always the case. In this regard, the authors do not show the feasibility of the method on non-convex models such as neural networks, where the complexity of the hypothesis function and the need for large training data might make this approximation less effective.

2.4.3 Defences Against Data Poisoning

It has become clear that ML algorithms require some sort of defence against poisoning attacks. A considerable problem with many defences is that they are overly specialised, with very few general poisoning defences which are able to handle multiple poisoning attacks (e.g., indiscriminate and targeted attacks simultaneously) (Cinà et al., 2023).

Data Sanitisation

These types of defences aim to remove poisoning samples before training to mitigate the attack. The reasoning behind this strategy is that for poisoning samples to be effective, they should possess different characteristics to the clean samples, or else there would be minimal impact on the training process.

Data sanitisation techniques—as illustrated in Fig. 2.5(c)—use anomaly detectors to filter out training points that look suspicious (Cretu et al., 2008; Huang et al., 2011). Reject on Negative Impact (RONI) was one of the first implementations to be robust to some attacks, making use of a calculation of the impact of each training example to

discard examples which have a drastically negative impact on the classifier's accuracy. Nonetheless, it needs retraining for each data point, making it very computationally expensive (Nelson et al., 2008). One possible way to reduce this complexity is by means of influence functions (Koh and Liang, 2017), so as to identify the most dangerous—likely to be influenced—samples in the training set that would need manual inspection. On the contrary, k-nearest neighbours labelling is a form of clustering that uses samples adjacent to a possible malicious point, to ascertain whether it is malicious—and if so, relabelling this introduced data point. This defence is limited when subsets of maliciously introduced data points are close (Paudice et al., 2018b), even if that subset is far from the rest of the data. In this sense, this detection mechanism is highly sensitive to the training data and might be prone to overfitting (Koh et al., 2022).

Another possible defence mechanism is anomaly detection, where an outlier detector (Hodge and Austin, 2004) is deployed against optimal poisoning attacks. This approach considers the whole ML pipeline, and uses pre-filtering with distance-based outlier detection (Paudice et al., 2018a). However, this defence works if the data used to train the outlier detector is *trusted*. If this assumption is violated, the outlier detector can be also poisoned. In these cases, defending is even harder since the adversary can be free to significantly change the underlying data distribution. Furthermore, the algorithm is only capable of detecting and removing poisoning points that are far from the genuine points of the corresponding class, which is not always the case for stealthy attackers (Steinhardt et al., 2017; Paudice et al., 2018a; Koh et al., 2022). Overall, these methods are less sensitive and might tend to underfit, because they make strong parametric assumptions (Koh et al., 2022). Label sanitisation has also been proposed as a mechanism to identify and relabel suspicious training points (Paudice et al., 2018b; Zhang et al., 2018). However, this strategy can fail when the poisoning points "collude" (Muñoz-González et al., 2019b). Finally, Diakonikolas et al. (2019) propose SEVER, a robust meta-algorithm based on singular value decomposition, capable of mitigating some poisoning attacks; and Yang et al. (2022) identify effective poisoning points as isolated medoids—most centrally located examples—of each class, in the gradient space.

Robust Training

In contrast to data sanitisation, robust training approaches help mitigate poisoning attacks by attempting to develop a training algorithm which limits the effectiveness of malicious points by design rather than removing these points altogether. These methods require access to the training data, the learning algorithm, and the model's parameters and thus can only be used when the defender trains the model themselves.

- **Regularisation**: a strategy utilised to reduce the influence of poisoning points in the training process is by increasing the level of regularisation, as shown by Carnerero-Cano et al. (2020, 2021, 2023) and in Ch. 3 of this thesis. Regularisation adds a penalty term to the original loss function which shrinks the norm of the model's parameters. This strategy is particularly effective against indiscriminate attacks. Carnerero-Cano et al. (2020, 2021, 2023) also describe the trade-off between accuracy and robustness, by means of a trusted validation set that guides the learning of the optimal regularisation hyperparameter that increases the validation accuracy.

- **Bagging and aggregation methods**: bagging refers to an ensemble learning protocol that trains sub-models on the sub-sampled sub-training sets and makes predictions by a majority voting mechanism, which is a commonly used method to avoid overfitting (Breiman, 1996). The defender can then train a model on each of the subsets, building an ensemble. The motivation behind this is that, to poison all small models simultaneously, many poisoning points would be required. The ensemble of models can then be combined to provide robustness (Biggio et al., 2011a; Jia et al., 2021; Chen et al., 2022). On the other hand, deep partition aggregation (Levine and Feizi, 2021) predicts through an aggregation of base classifiers trained on disjoint subsets of data, thus restricting its sensitivity to dataset distortions. Finite aggregation (Wang et al., 2022) first splits the training set into smaller disjoint subsets and then combines duplicates of them to build larger (but not disjoint) subsets for training base classifiers, which reduces the worst-case impacts of poisoning samples and slightly improves the ensemble's certification bounds compared to deep partition aggregation (Levine and Feizi, 2021).

- **Data augmentation**: Muñoz-González et al. (2019b) show empirically that for the same fraction of attack points, the effect of a poisoning attack reduces when

the number of training points increases, as the stability of the algorithm increases. Thus, data augmentation can help limit the attacker's ability to degrade the overall system's performance, leading to significant improvements in generalisation (Simard et al., 2003). However, data augmentation may not be as effective to defend against more targeted attacks.

- **Differential privacy**: while differential privacy was originally designed to formalise privacy—the output should not depend too much on any single individual's private data (Dwork et al., 2006; Abadi et al., 2016)—it also provides a defence against data poisoning. Differentially private learners are provably resistant to data poisoning attacks, with the protection degrading exponentially as the attacker poisons more points. However, specifically crafted attacks against differentially-private learners can be very effective (Ma et al., 2019).

- **Gradient shaping with differential privacy**: this technique focusses on the fact that most poisoning attacks share a common feature: they all modify the gradients used to train the model. Two key features of gradients that are different when computed in the presence of poisoning point are: (i) their ℓ_2-norms have much greater magnitudes than clean gradients; and (ii) their orientation differs from clean gradients. Hong et al. (2020) propose a generic poisoning defence which bounds gradient magnitudes and minimises diversion of orientation. They use differentially-private stochastic gradient descent (DP-SGD) to achieve this and show the defence to be effective against all attacks except for an unrealistically strong indiscriminate attack. Although DP-SGD defences do increase robustness, they come at a price: accuracy on the clean validation set can be significantly reduced.

- **Adversarial training**: in a manner similar to adversarial training in the domain of evasion attacks (Madry et al., 2018), Geiping et al. (2021b); Tao et al. (2021) describe methods to adversarially train a model against targeted poisoning attacks. In line with these defences, Liu et al. (2022) develop an adversarial training algorithm based on two components: an accuracy-friendly perturbation that is generated to maximally perturb samples without degrading the performance—to alleviate the local sharp regions introduced by the poisoning points—and a randomly varying noise component—to further smooth out the loss landscape.

Having reviewed the relevant literature in adversarial ML and data poisoning attacks and defences, in the next section we will provide an overview of the main technique developed in this thesis for analysing optimal poisoning attacks: bilevel optimisation.

2.5 Bilevel Optimisation

Bilevel programming problems (BLPPs) consist of two sub-problems called the upper-level and lower-level problems, where the upper-level (leader) problem must be solved subject to optimality of the lower-level (follower) problem (Mackay et al., 2019). BLPPs are thus hierarchical in the sense that their constraints are defined in part by a second parametric optimisation problem. In the case of adversarial ML, for instance, the outer level may correspond to the maximisation of the generalisation cost—tested on a holdout dataset—for the attacker, where the parameters of the generalisation error are subject to the minimisation of the training cost—evaluated on the available training dataset—for the defender. Additional constraints in the optimisation problem can be designed to ensure that the attacker evades detection.

Minimax problems are an example of bilevel programs where the upper-level objective equals the negative lower-level objective. Nevertheless, one aspect of bilevel problems is that they are generally not symmetric in terms of the two levels. The upper level decision maker usually has perfect knowledge of the lower level problem, while the lower level decision maker only observes the decisions of the leader and then optimises its own strategies. Interestingly, a limited knowledge about the follower's optimisation problem to the leader might conduct to bilevel optimisation problems involving uncertainties (Sinha et al., 2018).

The first studies of BLPPs were first carried out in the field of game theory (von Neumann, 1928) by the economist von Stackelberg (1934), to model firm dynamics. The strategic game described in his monograph came to be known as Stackelberg game that consists of a leader and a follower. In a Stackelberg game, two players (which cannot be interchanged) compete against each other, such as the leader makes the first move, and then the follower reacts optimally to the leader's action. If the leader—which knows ex ante that the follower observes its actions—wants to optimise its objective, then they need to anticipate the optimal response of the follower. A few decades later, the first mathematical model for BLPPs was provided by Bracken and McGill (1973).

Since then, there has been a steady growth in investigations and applications of bilevel optimisation in various practical problems (see (Dempe, 2002, 2003; Colson et al., 2005, 2007; Bard, 2013; Kalashnikov et al., 2015; Dempe et al., 2015; Sinha et al., 2018; Dempe, 2018) for an overview), such as economics (e.g., principal-agent problems, environmental economics), transportation (e.g., toll-setting problems, origin-destination matrix estimation, transportation planning), and engineering (e.g., optimal design, defence applications, chemical industry, inverse optimal control), amongst others (Dempe, 2003; Colson et al., 2007; Sinha et al., 2018). For example, some recent applications in defence include planning the prepositioning of defensive missile interceptors to counter an attack threat (Brown et al., 2005), interdicting nuclear weapons project (Brown et al., 2009), homeland security applications (Wein, 2009), and location problems (Aksen and Aras, 2012). The bilevel problem, while attacking, involves maximising the damage caused to the opponent by taking into account the optimal reactions of the opponent. Conversely, while defending, the bilevel problem involves minimising the maximum damage that an attacker can cause (Sinha et al., 2018).

In ML, many problems can be formulated as bilevel programs; for example, hyper-parameter optimisation (Bengio, 2000a; Foo et al., 2008; Domke, 2012; Franceschi et al., 2018), GANs training (Goodfellow et al., 2014a), meta-learning (Franceschi et al., 2018), and neural architecture search (Zoph and Le, 2017; Miikkulainen et al., 2019). If we take inspiration from the defence applications, these techniques are of great interest in the scope of this thesis because they appear in a natural way in adversarial ML problems, as they are able to dynamically model the interactions of attacker and defender by means of the problem constraints. In this sense, as mentioned in Subsect. 2.4.2, optimal poisoning strategies based on bilevel optimisation have been proposed in the literature to determine the maximum damage an adversary can perpetrate if they have perfect knowledge about the system, where the upper level represents the attacker and the lower level, the defender (Biggio et al., 2012; Mei and Zhu, 2015; Muñoz-González et al., 2017).

Let us now delve into the mathematical details of the problem. The general formulation of a continuous BLPP is composed of two nested optimisation problems (Dempe, 2003),

where the upper level is defined in the feasible closed set $\mathcal{X} \subseteq \mathbb{R}^n$ as:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X}} \quad & F(\mathbf{x}, \mathbf{y}(\mathbf{x})), \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x}, \mathbf{y}(\mathbf{x})) \preceq \mathbf{0}_k, \\ & \mathbf{H}(\mathbf{x}, \mathbf{y}(\mathbf{x})) = \mathbf{0}_l, \\ & \mathbf{y}(\mathbf{x}) \in \Psi(\mathbf{x}), \end{aligned} \tag{2.1}$$

where $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the upper (or outer) objective function, $\mathbf{G} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ and $\mathbf{H} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$ are, respectively, the upper inequality and upper equality constraints, $\mathbf{G}(\mathbf{x}, \mathbf{y}(\mathbf{x})) = (G_1(\mathbf{x}, \mathbf{y}(\mathbf{x})), ..., G_k(\mathbf{x}, \mathbf{y}(\mathbf{x})))^\mathsf{T}$ and $\mathbf{H}(\mathbf{x}, \mathbf{y}(\mathbf{x})) = (H_1(\mathbf{x}, \mathbf{y}(\mathbf{x})), ..., H_l(\mathbf{x}, \mathbf{y}(\mathbf{x})))^\mathsf{T}$, and $\mathbf{y}$, for a fixed value of $\mathbf{x}$, belongs to the parameterised solution set mapping $\Psi(\mathbf{x})$ of the so-called lower level problem. The parameterised solution set mapping is defined as:

$$\Psi(\mathbf{x}) := \{\mathbf{y} \in \mathcal{Y}(\mathbf{x}) \cap \mathcal{T} : f(\mathbf{x}, \mathbf{y}) \leq \varphi(\mathbf{x})\}, \tag{2.2}$$

where $\mathcal{Y}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ is the feasible set mapping, $\mathcal{T} \subseteq \mathbb{R}^m$ is a closed set, and $\varphi(\mathbf{x})$ is the optimal value function. The lower level problem is defined through the optimal value function as:

$$\begin{aligned} \varphi(\mathbf{x}) := \min_{\mathbf{y}\in\mathcal{T}} \quad & f(\mathbf{x}, \mathbf{y}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \mathbf{y}) \preceq \mathbf{0}_p, \\ & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}_q, \end{aligned} \tag{2.3}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the lower (or inner) objective function, and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ and $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$ are, respectively, the lower inequality and lower equality constraints. Furthermore, let the feasible set mapping $\mathcal{Y}(\mathbf{x})$ be:

$$\mathcal{Y}(\mathbf{x}) := \{\mathbf{y} : \mathbf{g}(\mathbf{x}, \mathbf{y}) \preceq \mathbf{0}_p, \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}_q\}. \tag{2.4}$$

The aim of the BLPP is then to implicitly determine the vector $\mathbf{x}$ describing the "environmental data" for the lower level problem which is the optimal one in the sense that the function F is minimised subject to some inequality and equality constraints (Dempe, 2003).

The relaxed version of the problem (2.1)-(2.3) is defined as:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X},\mathbf{y}} \quad & F(\mathbf{x},\mathbf{y}), \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x},\mathbf{y}) \preceq \mathbf{0}_k, \\ & \mathbf{H}(\mathbf{x},\mathbf{y}) = \mathbf{0}_l, \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \preceq \mathbf{0}_p, \\ & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}_q, \end{aligned} \tag{2.5}$$

and its optimal value is a lower bound for the optimal value of (2.1)-(2.3).

Even if all objectives and constraints are linear, bilevel problems—including the verification of local optimality for a feasible solution—have been shown to be non-convex, non-differentiable (Clark and Westerberg, 1983), and strongly NP-hard (Hansen et al., 1992; Vicente et al., 1994). This makes the computation of an optimal solution a challenging task. Moreover, in the presence of upper level constraints, the feasible set of the BLPP can be disconnected or even empty (Vicente and Calamai, 1994). The work of Edmunds and Bard (1991) shows a short description of the conditions under which this feasible set is compact and the BLPP has an optimal solution. Due to the difficulty in obtaining exact solutions, most work has focussed on restricted settings, considering linear, quadratic, and convex functions. In this thesis, we focus on obtaining local solutions in the convex and (in the case of deep learning) non-convex, differentiable settings.

To solve the BLLP (2.1)-(2.3)—or to formulate (necessary or sufficient) optimality conditions—one of the most usual approaches is to transform it into a single-level optimisation problem (Dempe et al., 2015), which is solved to arrive at a "satisficing"[7] instead of an optimal solution (Sinha et al., 2018). However, an important challenge is that BLPPs are non-convex optimisation problems. Due to the inherent difficulties for solving such problems globally, many researchers focus their investigations on *gradient-based methods* for computing stationary (local optima) solutions. In this sense, using gradient information provides drastic speed-ups over derivative-free optimisation methods[8] when

[7] A "satisficing" solution corresponds to an approximate Stackelberg strategy, which represents a reasonable compromise between the leader and the follower (Lai, 1996).

[8] Derivative-free optimisation techniques aim to optimise an objective (unknown) function $f(\mathbf{x})$: the algorithm may query the value $f(\mathbf{x})$ for a point $\mathbf{x}$, but it does not obtain gradient information, and in particular it does not make any assumptions on the analytic form of $f(\mathbf{x})$. In this regard, the optimisation procedure is dependent on the number of available queries to the unknown function (Loshchilov and Glasmachers, 2019).

we have high-dimensional parameters to optimise (Nesterov, 2013). Given that a point is considered feasible only if it is lower level optimal, finding a feasible direction along which the upper-level objective is optimised can be quite challenging. In this respect, the general assumptions are such that the objective functions are differentiable, and that their first and second order derivatives are uniformly continuous (Maclaurin et al., 2015; Pedregosa, 2016; Mei and Zhu, 2015). A naive method consists in applying simultaneous gradient descent, which updates $\mathbf{x}$ using $\nabla_{\mathbf{x}} F$ and $\mathbf{y}$ using $\nabla_{\mathbf{y}} f$. However, simultaneous gradient descent often gives incorrect solutions as it fails to account for the dependence of $\mathbf{y}$ on $\mathbf{x}$. Consider the relatively common situation where F does not have an explicit dependence on $\mathbf{x}$, so that $\nabla_{\mathbf{x}} F = \mathbf{0}_n$ is always true and hence $\mathbf{x}$ is never updated (Mackay et al., 2019). Then, applying the chain rule of differentiation, we have, whenever $\nabla_{\mathbf{x}} \mathbf{y}(\mathbf{x})$ is well defined:

$$\nabla_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}(\mathbf{x})) = \nabla_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) + (\nabla_{\mathbf{x}} \mathbf{y}(\mathbf{x}))^{\mathsf{T}} \nabla_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}), \tag{2.6}$$

where the functions are evaluated at the current iterate. To resolve Eq. (2.6), researchers have investigated ways to approximate the gradient of the upper level objective (Kolstad and Lasdon, 1990) as well as considered formulation of auxiliary programs (Savard and Gauvin, 1994) to determine the direction of descent.

In the adversarial ML community, previous studies have exploited gradient-based optimisation, along with the idea of implicit differentiation, to solve the bilevel problem of poisoning. Considering that the objective functions are convex and sufficiently regular, it is possible to replace the inner optimisation problem with its stationarity Karush-Kuhn-Tucker (KKT) conditions,[9] so as to derive an implicit equation for the gradient (Biggio et al., 2012; Koh and Liang, 2017; Mei and Zhu, 2015; Xiao et al., 2015; Muñoz-González et al., 2017; Koh et al., 2022). This method to solve the bilevel optimisation will be explored further in this thesis.

Alternative methods proposed in the literature to solve non-linear continuous BLPPs include (Sinha et al., 2018; Vicente and Calamai, 1994):

- **Branch-and-bound methods**: are applied to convex and regular bilevel programs. Although they are associated with large computational efforts—they constitute an

[9] In mathematical optimisation, the KKT conditions are first-order derivative tests necessary conditions for a solution in a non-linear program to be optimal, provided that some regularity conditions are satisfied (Karush, 1939; Kuhn and Tucker, 1951). Allowing inequality constraints, the KKT approach to non-linear programming generalises the method of Lagrange multipliers (Boyd and Vandenberghe, 2004).

exponentially slow algorithm with the number of integer variables—they are capable of computing global optima. Several approaches exploit the complementarity between the multipliers and the slack variables that arises from the KKT conditions of the lower level problem.

- **Penalty function methods**: some of these techniques can also be classified as gradient-based algorithms. The BLPP is handled by solving a series of unconstrained optimisation problems. The unconstrained problem is generated by adding a penalty term that measures the extent of violation of the constraints. The penalty term often requires a parameter and takes the value zero for feasible points and positive (minimisation) for infeasible point. However, they are generally limited to computing local optima.

- **Trust-region methods**: are iterative algorithms that approximate a certain region of the objective function with a model function around the current iterate. The region—usually defined as a p-norm ball centred at the iterate—is expanded if the approximation is good, otherwise it is contracted. These techniques can achieve an accurate global solution.

Finally, for highly complex or large scale bilevel problems, classic methods often fail due to difficulties such as non-linearity and discreteness. Under such circumstances, evolutionary approaches, though computationally demanding, can be alternative tools to offset some of these obstacles and lead to an approximate optimal solution (Sinha et al., 2018). In addition, the incorporation of a good member from one population to the population in the vicinity can boost convergence. For example, when it is hard to approximate the entire model with a small set of sample points, researchers often resort to iterative metamodelling-based methods (Sinha et al., 2013), where the actual model is approximated locally during iterations. A review of evolutionary approaches to solve BLPPs is presented by Sinha et al. (2018). Nonetheless, given that in this work our goal is to efficiently craft poisoning samples, with limited computational resources, in this case we will make use of classic approaches—specifically, gradient-based methods.

2.6 Conclusions

In this chapter, we have reviewed the state of the art of adversarial ML. Specifically, we have discussed the core security implications of the classical assumptions (IID samples) for ML models, when they are violated in adversarial environments; reviewed the threat model and the main kind of attacks we shall tackle in this thesis, i.e., poisoning attacks; and explained the main technique investigated throughout this thesis for analysing the security of ML algorithms, that is, bilevel optimisation, highlighting the convenience of gradient-based techniques to examine worst-case scenarios for the assurance evaluation of ML systems.

Thus, solving a bilevel optimisation problem is expected to enhance the effectiveness of poisoning attacks and their ability to evade defensive mechanisms. Specifically, the bilevel strategy currently stands as the state-of-the-art technique that enables crafting attacks when training from scratch. On the contrary, alternative heuristic methods, such as feature collision, have been proven to be entirely ineffective when the feature extractor is modified during training (Cinà et al., 2023).

However, the approach remains too computationally demanding to attack modern model architectures such as foundation models (Bommasani et al., 2021),[10] where the number of parameters is on the order of millions or billions. Consequently, improving the complexity of the bilevel problems to model poisoning attacks against foundation models—out of the scope of this thesis—is a relevant open change for ensuring a scalable evaluation of modern model architectures against such attacks (Cinà et al., 2023).

Equipped with the background discussed in this chapter, we are ready to move on to the technical contributions of this thesis. We shall start in Ch. 3 by investigating novel indiscriminate poisoning attacks that consider the effect of the hyperparameters in the model.

[10] A foundation model is any large ML model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks; some examples include BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020), and DALL-E (Ramesh et al., 2021). From a architectural point of view, foundation models are based on DNNs (Bommasani et al., 2021).

Chapter 3

Hyperparameter Learning under Data Poisoning

Synopsis

We show that current approaches, which typically assume that hyperparameters remain constant, lead to an overly pessimistic view of the algorithms' robustness and of the impact of regularisation. We propose a novel optimal attack formulation that considers the effect of the attack on the hyperparameters and models the attack as a *multiobjective* bilevel optimisation problem. This allows to formulate optimal attacks, learn hyperparameters and evaluate robustness under worst-case conditions. We apply this attack formulation to several ML classifiers using L_2 and L_1 regularisation. Our evaluation on multiple datasets shows that choosing an "a priori" constant value for the regularisation hyperparameter can be detrimental to the performance of the algorithms. This confirms the limitations of previous strategies and evidences the benefits of using L_2 and L_1 regularisation to dampen the effect of poisoning attacks, when hyperparameters are learnt using a small trusted dataset.

3.1 Introduction

This thesis proposes novel optimal indiscriminate attacks and corresponding defences considering the different stages of the supervised learning pipeline. In particular, optimal attacks can be formulated as bilevel optimisation problems and help assess their robustness in worst-case scenarios. We will begin our road map by analysing the interactions

between the optimal attack and the hyperparameters of the model.

In this chapter, we shall answer the following research questions stated in Ch. 1:

- **RQ3**: Can we understand the impact of poisoning attacks in the hyperparameters of the model? In particular, can we understand the different trade-offs between accuracy and robustness according to the value of the hyperparameters
- **RQ4**: Can a small trusted validation set help learn the correct hyperparameters under data poisoning?
- **RQ5**: Can regularisation help mitigate poisoning attacks?

Specifically, we show that current approaches, which typically assume that hyperparameters remain constant, lead to an overly pessimistic view of the algorithms' robustness and of the impact of regularisation. We provide a comprehensive analysis of the influence of the hyperparameters on the effect of poisoning attacks when using different regularisation techniques, including L_2 and L_1 regularisation. We also propose a more general optimal indiscriminate poisoning attack formulation to test worst-case scenarios against ML algorithms that contain hyperparameters. For this, we model the attack as a *multiobjective bilevel optimisation problem*, where the outer objective includes both the learning of the poisoning points and that of the hyperparameters, while the inner problem involves learning the model's parameters. This attack formulation allows us to model an adversary aware not only of the training algorithm, but also of the procedure used to select the model's hyperparameters. Thus, this formulation considers a more realistic attacker and assesses in a more comprehensive way the robustness of the algorithms to poisoning attacks in worst-case scenarios. In scenarios where the attacker is aware of the dataset used to learn the model's hyperparameters and aims to maximise the overall error, the outer objective can be modelled as a *minimax* problem.

We use *hypergradient* (i.e., the gradient in the outer problem (Maclaurin et al., 2015; Franceschi et al., 2017, 2018; Grazzi et al., 2020)) descent/ascent to solve the multiobjective bilevel optimisation problem. As the computation of the exact hypergradients can be computationally expensive, especially for neural networks, we use RMD (Domke, 2012; Maclaurin et al., 2015; Franceschi et al., 2017; Muñoz-González et al., 2017; Franceschi et al., 2018; Grazzi et al., 2020) to approximate the hypergradients. We conduct an exhaustive experimental analysis on logistic regression (LR) and DNNs, using different

datasets including MNIST (LeCun et al., 1998), Fashion-MNIST (FMNIST) (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al., 2009), and attacks with both small and large fractions of poisoning points.

We show that choosing (*a priori*) a constant value for the regularisation hyperparameter, λ, can be detrimental: if the value is too high it damages accuracy (i.e., it produces underfitting when there is no attack), if the value is too low it damages robustness (the algorithm is more brittle in the presence of an adversary). In contrast, selecting λ appropriately by, for example, using a small trusted validation set, provides both accuracy and robustness regardless of the presence or absence of poisoning points in the training dataset and of the attack strength. Our empirical evaluation also reveals that the value of the regularisation hyperparameter increases with the number of poisoning points injected in the training set. The algorithm automatically tries to compensate the negative effect of the poisoning points by increasing the strength of the regularisation term. For DNNs, we show that the attack can have a more pronounced effect in the later layers of the network, and that the use of different regularisation hyperparameters for the different layers in the DNN can be beneficial to mitigate the impact of data poisoning. In the case of embedded feature selection methods, we confirm the stabilising effect of regularisation against poisoning.

3.2 Related Work

Previous attacks in the literature (Biggio et al., 2012; Mei and Zhu, 2015; Xiao et al., 2015; Muñoz-González et al., 2017; Koh et al., 2022) target algorithms that have hyperparameters, but the hyperparameters are considered constant regardless of the fraction of poisoning points injected in the training dataset. This can provide a misleading analysis of the robustness of the algorithms to such attacks, as the value of the hyperparameters can change depending on the type and strength of the attack. For example, Xiao et al. (2015) presented a poisoning attack against embedded feature selection methods, including L_1, L_2 and *elastic-net* regularisation. Their results show that the attacker can completely control the selection of the features to significantly increase the overall test error of linear classifiers. However, they assume a constant regularisation hyperparameter regardless of the attack considered. We show that this approach provides overly pessimistic results on the ML algorithms' robustness to poisoning attacks.

3.3 General Optimal Poisoning Attacks

In data poisoning attacks the attacker can tamper with a fraction of the training set to manipulate the behaviour of the learning algorithm (Barreno et al., 2006, 2010). We assume that the attacker can arbitrarily manipulate all the features and the label of the injected poisoning points, provided that the resulting points are within a feasible domain of valid data points. We consider PK attacks, i.e., the attacker knows everything about the target system, including the training data, the feature representation, the loss function, the ML model, and the defence (if applicable) used by the victim. Although unrealistic in most cases, these assumptions are needed to analyse the robustness of the ML algorithms in worst-case scenarios to attacks of different strengths.

3.3.1 Problem Formulation

In line with most literature on poisoning attacks we consider ML classifiers. Then, in a classification task, given the input space $\mathcal{X} \subseteq \mathbb{R}^m$ and the discrete label space, $\mathcal{Y} \subseteq \mathbb{Z}_c$, where c is the number of classes, the learner aims to estimate the mapping $f : \mathcal{X} \to \mathcal{Y}$. Given a training set $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}_{\text{tr}_i}, y_{\text{tr}_i})\}_{i=1}^{n_{\text{tr}}}$ with n_{tr} IID samples drawn from the underlying probability distribution $p(\mathcal{X}, \mathcal{Y})$, we can estimate f with a model $\mathcal{M}$: $\mathbb{R}^{n_{\text{tr}} \times m} \to \mathbb{R}^{n_{\text{tr}} \times c}$ trained by minimising an objective function $\mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{\Lambda}, \mathbf{w}) : \mathbb{R}^{n_{\text{tr}} \times m} \times \mathbb{Z}_c^{n_{\text{tr}}} \to \mathbb{R}$ w.r.t. its parameters,[1] $\mathbf{w} \in \mathbb{R}^d$, given a set of hyperparameters $\mathbf{\Lambda} \in \mathbb{R}^h$.

We use gradient-based algorithms to optimise the performance of the model on a clean validation set with respect to the hyperparameters (Larsen et al., 1996; Bengio, 2000b; Foo et al., 2008; Franceschi et al., 2017, 2018) and poisoning points (Biggio et al., 2012; Xiao et al., 2015; Muñoz-González et al., 2017). Thus, we assume that the defender has access to a small validation dataset $\mathcal{D}_{\text{val}} = \{(\mathbf{x}_{\text{val}_j}, y_{\text{val}_j})\}_{j=1}^{n_{\text{val}}}$ with n_{val} trusted data points, representative of the ground-truth underlying data distribution. In practice, it is not uncommon to have access to a limited clean set, for example, because the integrity of a small set of data sources can be ascertained.[2] This small clean dataset is held out for the optimisation of the hyperparameters (and the poisoning points, as we describe later). Then, as proposed by Foo et al. (2008), the model's hyperparameters can be learnt by

[1] As Maclaurin et al. (2015) we use parameters to denote "parameters that are just parameters and not hyperparameters".

[2] Note that if the quality of the trusted data is limited, the model's performance can be limited as well.

solving the following bilevel optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{\Lambda}' \in \Phi(\boldsymbol{\Lambda})} \quad & \mathcal{L}(\mathcal{D}_{\text{val}}, \mathbf{w}^\star) \\ \text{s.t.} \quad & \mathbf{w}^\star \in \arg\min_{\mathbf{w} \in \mathcal{W}} \quad \mathcal{L}\left(\mathcal{D}_{\text{tr}}, \boldsymbol{\Lambda}', \mathbf{w}\right), \end{aligned} \tag{3.1}$$

where $\Phi(\boldsymbol{\Lambda})$ represents the feasible domain set for the hyperparameters $\boldsymbol{\Lambda}$. The use of this approach to select the model's hyperparameters has some advantages compared to other selection methods. Cross-validation-based approaches (Hastie et al., 2009) require retraining the model multiple times over different training and validation set splits, making it computationally very demanding when the number of hyperparameters is large and training the learning algorithm is expensive. Grid search techniques (Feurer and Hutter, 2019) also rely on a separate validation set to select the hyperparameters. However, the exploration of all the hyperparameters values considered in the grid also requires to train the learning algorithm many times, which can be computationally demanding, especially as the number of hyperparameters in the model grows. This can be alleviated using more guided search techniques, such as Bayesian optimisation, but the exploration of each combination of hyperparameters requires training the algorithms from scratch multiple times and the performance and scalability with the number of hyperparameters is reduced.

In contrast, solving the bilevel optimisation problem in Eq. (3.1), with gradient-based techniques, is computationally more efficient than previous approaches when using approximate techniques to estimate the hypergradients in the outer objective (Domke, 2012; Pedregosa, 2016; Franceschi et al., 2017, 2018; Muñoz-González et al., 2017). In this case, the computation of these hypergradients does not require training the learning algorithm (in the inner objective) fully, but only for a reduced number of epochs. This approach is more scalable, especially when the number of hyperparameters is large. On the downside, gradient-based techniques to solve Eq. (3.1) do not guarantee to find the global optimum for the outer objective but only a local one. However, this problem can be mitigated with multiple restarts.

In a poisoning attack, the adversary aims to inject a set of n_{p} poisoning data points, $\mathcal{D}_{\text{p}} = \{(\mathbf{x}_{\text{p}_k}, y_{\text{p}_k})\}_{k=1}^{n_{\text{p}}}$, in the training set to maximise some arbitrary objective, $\mathcal{A} : \mathbb{R}^{n_{\text{target}} \times m} \times \mathbb{Z}_c^{n_{\text{target}}} \to \mathbb{R}$, evaluated on a set of target data points $\mathcal{D}_{\text{target}}$. As described by Muñoz-González et al. (2017) different attack scenarios can be considered depending on

both the set of target data points and the attacker's objective, including indiscriminate and targeted attacks. To allow for the learning of the hyperparameters we therefore propose to formulate the attacker's problem as a *multiobjective* bilevel optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{\Lambda}' \in \Phi(\boldsymbol{\Lambda})} \mathcal{L}(\mathcal{D}_{\text{val}}, \mathbf{w}^\star), \max_{\mathcal{D}'_{\text{p}} \in \Phi(\mathcal{D}_{\text{p}})} \mathcal{A}(\mathcal{D}_{\text{target}}, \mathbf{w}^\star) \\ \text{s.t.} \quad \mathbf{w}^\star \in \arg\min_{\mathbf{w} \in \mathcal{W}} \mathcal{L}\left(\mathcal{D}'_{\text{tr}}, \boldsymbol{\Lambda}', \mathbf{w}\right), \end{aligned} \tag{3.2}$$

where $\mathcal{D}'_{\text{tr}} = \mathcal{D}_{\text{tr}} \cup \mathcal{D}'_{\text{p}}$ is the poisoned dataset and $\Phi(\mathcal{D}_{\text{p}})$ is the feasible domain for the attacker.

From the general formulation in Eq. (3.2) it is clear that the poisoning points in $\mathcal{D}'_{\text{tr}}$ have an effect not only on the parameters of the classifier (in the inner problem), but also on its hyperparameters (in the outer objective for the defender). Previous studies have neglected the effect of the hyperparameters in the problem for the attacker, e.g., the regularisation hyperparameter for the loss function for SVMs (Biggio et al., 2012) or for embedded feature selection methods (Xiao et al., 2015). This can overestimate the adversary's capabilities to influence the learning algorithm, as we show in the synthetic experiment in Fig. 3.1.

Our novel attack formulation in Eq. (3.2) allows to model a wide variety of attack scenarios, depending on the attacker's objective and the combinations between the target, validation and training sets. However, in this thesis we focus on analysing *worst-case scenarios for indiscriminate poisoning attacks*, where the attacker, having perfect knowledge, aims to increase the overall classification error in the target algorithm. These settings have been commonly used in most of the related work on poisoning attacks using bilevel optimisation (Biggio et al., 2012; Mei and Zhu, 2015; Xiao et al., 2015; Muñoz-González et al., 2017; Carnerero-Cano et al., 2020). To achieve such a goal, the attacker aims to maximise the loss evaluated on a separate validation set, i.e., $\mathcal{A}(\mathcal{D}_{\text{target}}, \mathbf{w}^\star) = \mathcal{L}(\mathcal{D}_{\text{val}},\ \mathbf{w}^\star)$: $\mathbb{R}^{n_{\text{val}} \times m} \times \mathbb{Z}_c^{n_{\text{val}}} \to \mathbb{R}$. In our case, where the attacker is also aware of the effect of the hyperparameters in the performance of the algorithm, $\mathcal{D}_{\text{val}}$ is the same as the validation dataset used by the defender, to maximise the overall error not only compromising the learning of the model's parameters, but also the selection (or learning) of its hyperparameters. Then, the attacker's problem can be formulated a bilevel

optimisation problem where the outer objective is a *minimax* problem:

$$\begin{aligned} \min_{\boldsymbol{\Lambda}' \in \Phi(\boldsymbol{\Lambda})} \max_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} & \mathcal{L}(\mathcal{D}_{\mathrm{val}}, \mathbf{w}^\star) \\ \text{s.t.} \quad \mathbf{w}^\star \in \arg\min_{\mathbf{w} \in \mathcal{W}} & \mathcal{L}\left(\mathcal{D}'_{\mathrm{tr}}, \boldsymbol{\Lambda}', \mathbf{w}\right). \end{aligned} \tag{3.3}$$

In this formulation, in the outer problem, there is an implicit dependency of both the hyperparameters, $\boldsymbol{\Lambda}$, and the poisoning points, $\mathcal{D}_{\mathrm{p}}$, on the parameters of the model learnt in the inner optimisation problem, $\mathbf{w}^\star$. We can also observe that the value of the poisoning points has an effect on the learning of both $\mathbf{w}$ and $\boldsymbol{\Lambda}$ in the inner and outer objectives respectively.

This formulation is compatible with grid-search-based approaches, which select the hyperparameters using a separate validation dataset. However, it is computationally infeasible to solve the problem for the attacker using these techniques, as the number of variables to be learnt in the outer objective, i.e., the model's hyperparameters and the value of the features for all the poisoning points, is very large. On the other hand, cross-validation uses the same dataset for creating the different training and validation splits. Thus, in that case the learner cannot benefit from the trusted dataset and, both the training and validation datasets would contain poisoning points across all splits. In our case, the learner uses the trusted set for validation aiming to mitigate the effect of the poisoning attack by the selection of appropriate hyperparameters.

Our experiments show that this can be a good approach in some cases, for example, when using regularisation to increase the stability of the learning algorithm, and helps mitigate the attack. Of course, more specialised algorithms can be devised to make a different use of the trusted set of data points—e.g., data hypercleaning (Franceschi et al., 2017). However, our intention here is to show that the existence of a trusted dataset can be helpful to reduce the impact of poisoning attacks just by using standard techniques to increase the stability of the algorithm, such as regularisation, and learning the model's hyperparameters appropriately. Our attack formulation allows us to characterise the worst-case performance under such assumptions. Thus, our findings provide ML practitioners a methodology to better use their trusted data points to mitigate poisoning attacks without requiring specialised knowledge or algorithms, but using techniques commonly used for training ML algorithms, as is the case of regularisation.

3.3.2 Solving General Optimal Poisoning Attacks

Solving the multiobjective bilevel optimisation problems in Eq. (3.2) and Eq. (3.3) is strongly NP-Hard (Bard, 2013) and, even if the inner problem is convex, the bilevel problem is, in general, non-convex. However, it is possible to use gradient-based approaches to obtain (possibly) suboptimal solutions, i.e., finding local optima for the problem in Eq. (3.2) and saddle points for the minimax problem in Eq. (3.3). In this thesis we focus on the solution to Eq. (3.3), which we use in our experiments to show the robustness of L_2 regularisation to indiscriminate poisoning attacks. The solution to Eq. (3.2) follows a similar procedure.

Similar to Biggio et al. (2012); Mei and Zhu (2015); Xiao et al. (2015); Muñoz-González et al. (2017), we assume that the label of the poisoning points is set a priori, so the attacker just needs to learn the features for the poisoning points, $\mathbf{X}_\mathrm{p}$. For clarity, in the following description we use $\mathcal{A}$ (which does not explicitly depend on the poisoning points or the hyperparameters, but implicitly through the parameters) to denote the loss function evaluated on $\mathcal{D}_\mathrm{val}$ in the outer objective, i.e., $\mathcal{L}(\mathcal{D}_\mathrm{val}, \mathbf{w}^\star)$, and $\mathcal{L}$ to refer to the loss function evaluated on $\mathcal{D}'_\mathrm{tr}$ in the inner objective, $\mathcal{L}(\mathcal{D}'_\mathrm{tr}, \mathbf{\Lambda}, \mathbf{w}^\star)$. Both are evaluated on $\mathbf{w}^\star$, the parameters obtained when solving the inner optimisation problem.

To compute the hypergradients for the outer objective, we assume that the first and second derivatives of the loss function, $\mathcal{L}$, are Lipschitz-continuous functions. We can then compute the hypergradients applying the chain rule, so that $\nabla_{\mathbf{X}_\mathrm{p}}\mathcal{A} = (d\mathbf{w}^\star/d\mathbf{X}_\mathrm{p})^\mathsf{T} \nabla_\mathbf{w}\mathcal{L}$.[3] To compute the implicit derivative, $d\mathbf{w}^\star/d\mathbf{X}_\mathrm{p}$, we can leverage the stationarity (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951) in the inner problem, i.e., $\nabla_\mathbf{w}\mathcal{L} = \mathbf{0}$, and apply the implicit function theorem (Mei and Zhu, 2015; Pedregosa, 2016; Koh and Liang, 2017), so that $\nabla_{\mathbf{X}_\mathrm{p}}\nabla_\mathbf{w}\mathcal{L} + (d\mathbf{w}^\star/d\mathbf{X}_\mathrm{p})^\mathsf{T} \nabla_\mathbf{w}\mathcal{L} = \mathbf{0}$.[3] Then, the hypergradients can be computed as

$$\nabla_{\mathbf{X}_\mathrm{p}}\mathcal{A} = -\left(\nabla_{\mathbf{X}_\mathrm{p}}\nabla_\mathbf{w}\mathcal{L}\right)^\mathsf{T} \left(\nabla^2_\mathbf{w}\mathcal{L}\right)^{-1} \nabla_\mathbf{w}\mathcal{A}, \tag{3.4}$$

where we assume that the Hessian $\nabla^2_\mathbf{w}\mathcal{L}$ is not singular. Brute-force computation of Eq. (3.4) requires inverting the Hessian, which scales in time as $\mathcal{O}(d^3)$ and in space as $\mathcal{O}(d^2)$—where d is the number of parameters. However, as Foo et al. (2008); Pedregosa (2016), we can rearrange the terms in the second part of Eq. (3.4), solve the linear system:

[3] The expression for $\mathbf{\Lambda}$ is analogous.

$\left(\nabla^2_{\mathbf{w}}\mathcal{L}\right)\mathbf{v} = \nabla_{\mathbf{w}}\mathcal{A}$, and compute $\nabla_{\mathbf{X}_\mathrm{p}}\mathcal{A} = -\left(\nabla_{\mathbf{X}_\mathrm{p}}\nabla_{\mathbf{w}}\mathcal{L}\right)^\mathsf{T}\mathbf{v}$.[3] The linear system can be efficiently solved by using Conjugate Gradient (CG) descent, as described by Foo et al. (2008). For this, let us assume that the inner problem is solved by an iterative algorithm that arrives at a local minima after T_{KKT} training iterations. After solving the linear system, the procedure scales in time $\mathcal{O}\left(\left(T_{\mathrm{KKT}} + \sqrt{\kappa}\right)d\right)$ and in space $\mathcal{O}(d)$ (Shewchuk, 1994), where κ is the condition number of the Hessian $\nabla^2_{\mathbf{w}}\mathcal{L}$. Moreover, the Hessian-vector products $\left(\nabla^2_{\mathbf{w}}\mathcal{L}\right)\mathbf{v}$ and $\left(\nabla_{\mathbf{X}_\mathrm{p}}\nabla_{\mathbf{w}}\mathcal{L}\right)^\mathsf{T}\mathbf{v}$ can be computed exactly and efficiently with the technique proposed in (Pearlmutter, 1994), thus avoiding the computation and storage of the Hessian, as follows:[3]

$$\begin{aligned}\left(\nabla^2_{\mathbf{w}}\mathcal{L}\right)\mathbf{v} &= \nabla_{\mathbf{w}}\left(\mathbf{v}^\mathsf{T}\nabla_{\mathbf{w}}\mathcal{L}\right),\\ \left(\nabla_{\mathbf{X}_\mathrm{p}}\nabla_{\mathbf{w}}\mathcal{L}\right)^\mathsf{T}\mathbf{v} &= \nabla_{\mathbf{X}_\mathrm{p}}\left(\mathbf{v}^\mathsf{T}\nabla_{\mathbf{w}}\mathcal{L}\right).\end{aligned} \tag{3.5}$$

The computation of the first and second expression above scales as $\mathcal{O}(d)$ and $\mathcal{O}(\max(d, n_\mathrm{p}m))$[4] respectively—where n_p denotes the number of poisoning points, each one containing m features—both in time and in space. An elegant aspect of this technique is that, for ML models optimised with gradient-based methods, the equations for evaluating the Hessian-vector products emulate closely those for standard forward and backward propagation. Hence, the application of existing automatic differentiation frameworks to compute this product is typically straightforward (Pearlmutter, 1994; Bishop, 2006).

However, approaches based on the implicit function theorem require training the whole learning algorithm to compute the hypergradient, i.e., until the stationarity conditions are met. This can be intractable for some learning algorithms such as deep networks, where the number of parameters is huge. To sidestep this problem, different techniques have been proposed to estimate the value of the hypergradients (Domke, 2012; Maclaurin et al., 2015; Pedregosa, 2016; Franceschi et al., 2017; Muñoz-González et al., 2017; Franceschi et al., 2018; Grazzi et al., 2020). These techniques do not require retraining the learning algorithm each time the hypergradient is computed. Instead, they estimate the hypergradient by truncating the learning in the inner problem to a reduced number of training iterations.

As described by Franceschi et al. (2017), we can think of the training algorithm (inner problem) as a discrete-time dynamical system, described by a sequence of states $\mathbf{s}^{(t)}\left(\mathbf{X}_\mathrm{p}, \mathbf{\Lambda}\right) \in \mathbb{R}^{d_\mathrm{s}}$, with $t = 1, \ldots, T$, where each state depends on model's parameters, the

[4] $\mathcal{O}(\max(d, h))$ for $\mathbf{\Lambda}$, where h is the number of hyperparameters.

accumulated gradients and/or the velocities, and the training data and hyperparameters. In this chapter, we focus on stochastic gradient descent (SGD), i.e., $\mathbf{s}^{(t)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right) = \mathbf{w}^{(t)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)$, so that each state of the sequence depends *only* on the previous state. We can therefore reformulate the bilevel problem in (3.3) as the constrained single-level optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{\Lambda}' \in \Phi(\boldsymbol{\Lambda})} \max_{\mathbf{X}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} & \mathcal{L}\left(\mathcal{D}_{\mathrm{val}}, \mathbf{w}^{(T)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)\right) \\ \text{s.t.} \quad & \mathbf{w}^{(t)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}'\right) = \mathbf{w}^{(t-1)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right) \\ & \qquad - \eta \nabla_{\mathbf{w}} \mathcal{L}\left(\mathcal{D}'_{\mathrm{tr}}, \boldsymbol{\Lambda}', \mathbf{w}^{(t-1)}\right), \\ & t = 1, \ldots, T, \end{aligned} \tag{3.6}$$

where η is the learning rate for SGD. Then, we estimate the hypergradients from the values of the parameters collected in the set of training states as[3]

$$\nabla_{\mathbf{X}_{\mathrm{p}}} \mathcal{A} = \left(\frac{d\mathbf{w}^{(T)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)}{d\mathbf{X}_{\mathrm{p}}}\right)^{\mathsf{T}} \nabla_{\mathbf{w}} \mathcal{A}, \tag{3.7}$$

where the bottleneck is, again, the computation of the implicit derivatives. Given the constraints in Eq. (3.6), it is obvious that the state $\mathbf{w}^{(t)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)$ depends on the poisoning points and hyperparameters both, directly by its expression, and indirectly through the previous state $\mathbf{w}^{(t-1)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)$. Then, by applying the chain rule we obtain[3]

$$\frac{d\mathbf{w}^{(t)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)}{d\mathbf{X}_{\mathrm{p}}} = \frac{\partial \mathbf{w}^{(t)}}{\partial \mathbf{X}_{\mathrm{p}}} + \frac{\partial \mathbf{w}^{(t)}}{\partial \mathbf{w}^{(t-1)}} \frac{d\mathbf{w}^{(t-1)}\left(\mathbf{X}_{\mathrm{p}}, \boldsymbol{\Lambda}\right)}{d\mathbf{X}_{\mathrm{p}}} \tag{3.8}$$

Then, from a reduced number of training iterations, $T \leq T_{\mathrm{KKT}}$ (which do not necessarily satisfy the stationarity conditions (Franceschi et al., 2017; Muñoz-González et al., 2017; Franceschi et al., 2018; Grazzi et al., 2020)), these expressions can be expanded, according to the updates of SGD (Domke, 2012; Franceschi et al., 2017), as follows:

$$\begin{aligned} \nabla_{\mathbf{X}_{\mathrm{p}}} \mathcal{A} &= \left(\frac{\partial \mathbf{w}^{(T)}}{\partial \mathbf{X}_{\mathrm{p}}} + \sum_{t=1}^{T-1} \left(\prod_{t'=t+1}^{T} \frac{\partial \mathbf{w}^{(t')}}{\partial \mathbf{w}^{(t'-1)}}\right) \frac{\partial \mathbf{w}^{(t)}}{\partial \mathbf{X}_{\mathrm{p}}}\right) \nabla_{\mathbf{w}} \mathcal{A}, \\ \nabla_{\boldsymbol{\Lambda}} \mathcal{A} &= \left(\frac{\partial \mathbf{w}^{(T)}}{\partial \boldsymbol{\Lambda}} + \sum_{t=1}^{T-1} \left(\prod_{t'=t+1}^{T} \frac{\partial \mathbf{w}^{(t')}}{\partial \mathbf{w}^{(t'-1)}}\right) \frac{\partial \mathbf{w}^{(t)}}{\partial \boldsymbol{\Lambda}}\right) \nabla_{\mathbf{w}} \mathcal{A}, \end{aligned} \tag{3.9}$$

where $\partial \mathbf{w}^{(t')} / \partial \mathbf{w}^{(t'-1)} = \mathbf{I}_d - \eta \nabla^2_{\mathbf{w}} \mathcal{L}$, $\partial \mathbf{w}^{(t)} / \partial \mathbf{X}_{\mathrm{p}} = -\eta \nabla_{\mathbf{X}_{\mathrm{p}}} \nabla_{\mathbf{w}} \mathcal{L}$, $\partial \mathbf{w}^{(t)} / \partial \boldsymbol{\Lambda} = -\eta \nabla_{\boldsymbol{\Lambda}} \nabla_{\mathbf{w}} \mathcal{L}$, and $\mathbf{I}_d$ is the identity matrix.

Method	Time Complexity	Space Complexity	Requires Stationarity
Exact	$\mathcal{O}\left(d^3\right)$	$\mathcal{O}\left(d^2\right)$	Yes
CG	$\mathcal{O}\left(\left(T_{\text{KKT}} + \sqrt{\kappa}\right) d\right)$	$\mathcal{O}(n_{\text{p}}m + h + d)$	Yes
RMD	$\mathcal{O}(Td)$	$\mathcal{O}(n_{\text{p}}m + h + Td)$	No
FMD	$\mathcal{O}((n_{\text{p}}m + h)Td)$	$\mathcal{O}((n_{\text{p}}m + h)d)$	No

Table 3.1: Complexity of state-of-the-art methods to compute the hypergradient for the poisoning points and hyperparameters, assuming that SGD is the solver for the inner problem.

Depending on the order to compute the different terms in Eq. (3.9), we can use two approaches to estimate the hypergradients: RMD and forward-mode differentiation (FMD) (Griewank and Walther, 2008; Franceschi et al., 2017). In the first case, RMD requires first to train the learning algorithm for T training iterations, i.e., to compute $\mathbf{w}^{(1)}$ to $\mathbf{w}^{(T)}$. Then, the hypergradients estimate is computed by reversing the steps followed by the learning algorithm from $\mathbf{w}^{(T)}$ down to $\mathbf{w}^{(1)}$. On the other hand, FMD computes the estimate of the hypergradients as the algorithm is trained, i.e., from $\mathbf{w}^{(1)}$ to $\mathbf{w}^{(T)}$ (i.e., the estimates can be computed in parallel with the training procedure).

To estimate the hypergradients, RMD requires to compute a forward and a backward pass through the set of states. In some cases, as in the works of Domke (2012); Franceschi et al. (2017, 2018), RMD requires to store all the information collected in the states in the forward pass.[5] In contrast, FMD just needs to do the forward computation. However, compared to RMD, the scalability of FMD depends heavily on the number of hyperparameters. As a practical example, consider training a neural network (including LR as a special case) with d weights, using classic iterative optimisation algorithms such as SGD. According to Eq. (3.9), RMD scales in time as $\mathcal{O}(Td)$ and in space as $\mathcal{O}(n_{\text{p}}m + h + Td)$, while FMD scales as $\mathcal{O}((n_{\text{p}}m + h)Td)$ and $\mathcal{O}((n_{\text{p}}m + h)d)$ in time and space respectively. Thus, the time complexity of RMD does not depend on the size of the poisoning points or hyperparameters. Then, for problems where the number of hyperparameters is large, as is the case for the poisoning attacks we introduced in this chapter, RMD is computationally more efficient to estimate the hypergradients. As mentioned before, it is also clear that RMD is more efficient compared to grid search, where the learning algorithms need to be trained from scratch for each combination of the hyperparameters' values explored in the grid.

[5]However, other RMD methods proposed in the literature do not require to store this information (Maclaurin et al., 2015; Muñoz-González et al., 2017).

Algorithm 3.1: RMD for computing the hypergradients w.r.t. the hyperparameters and poisoning points

Input: $\mathcal{M}$, $\mathcal{A}$, $\mathcal{L}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}'_{\text{tr}}$, $\mathbf{\Lambda}$, $\mathbf{w}^{(0)}$, T, η
Output: $\nabla_{\mathbf{X}_\text{p}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right)$, $\nabla_{\mathbf{\Lambda}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right)$

1: **for** $t = 0$ **to** $T-1$ **do**
2: $\quad \mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} \mathcal{L}\left(\mathcal{D}'_{\text{tr}}, \mathbf{\Lambda}, \mathbf{w}^{(t)}\right)$
3: **end for**
4: $\partial \mathbf{w}^{(T)} \leftarrow \nabla_{\mathbf{w}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right)$
5: $\partial \mathbf{X}_\text{p}^{(T)} \leftarrow \mathbf{0}$
6: $\partial \mathbf{\Lambda}^{(T)} \leftarrow \mathbf{0}$
7: **for** $t = T-1$ **down to** 0 **do**
8: $\quad \partial \mathbf{g}_{\text{X}_\text{p}} \leftarrow \left(\nabla_{\mathbf{X}_\text{p}} \nabla_{\mathbf{w}} \mathcal{L}\left(\mathcal{D}'_{\text{tr}}, \mathbf{\Lambda}, \mathbf{w}^{(t)}\right)\right)^{\mathsf{T}} \partial \mathbf{w}^{(t+1)}$
9: $\quad \partial \mathbf{g}_{\Lambda} \leftarrow \left(\nabla_{\mathbf{\Lambda}} \nabla_{\mathbf{w}} \mathcal{L}\left(\mathcal{D}'_{\text{tr}}, \mathbf{\Lambda}, \mathbf{w}^{(t)}\right)\right)^{\mathsf{T}} \partial \mathbf{w}^{(t+1)}$
10: $\quad \partial \mathbf{w}^{(t)} \leftarrow \left(\mathbf{I}_d - \eta \nabla^2_{\mathbf{w}} \mathcal{L}\left(\mathcal{D}'_{\text{tr}}, \mathbf{\Lambda}, \mathbf{w}^{(t)}\right)\right) \partial \mathbf{w}^{(t+1)}$
11: $\quad \partial \mathbf{X}_\text{p}^{(t)} \leftarrow \partial \mathbf{X}_\text{p}^{(t+1)} - \eta \partial \mathbf{g}_{\text{X}_\text{p}}$
12: $\quad \partial \mathbf{\Lambda}^{(t)} \leftarrow \partial \mathbf{\Lambda}^{(t+1)} - \eta \partial \mathbf{g}_{\Lambda}$
13: **end for**
14: $\nabla_{\mathbf{X}_\text{p}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right) \leftarrow \partial \mathbf{X}_\text{p}^{(0)}$
15: $\nabla_{\mathbf{\Lambda}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right) \leftarrow \partial \mathbf{\Lambda}^{(0)}$

Table 3.1 summarises the computational trade-offs between different state-of-the-art methods to compute the hypergradients. For the analysis of the convergence properties of the hypergradients, we refer the reader to (Grazzi et al., 2020), which studies and compares the convergence rate of techniques such as CG and RMD. From a practical perspective, the number of training iterations for the inner problem plays a crucial role in the convergence rate (Pedregosa, 2016; Franceschi et al., 2018; Grazzi et al., 2020), but can also cause overfitting in the outer objective (Franceschi et al., 2018).

Here we include the RMD algorithm (Alg. 3.1), which we use to compute the hypergradients estimate at the outer level problem (both for the features of the poisoning points (Line 14), and the hyperparameters (Line 15)). RMD requires first to train the learning algorithm for T training iterations (Lines 1-3). Then, the hypergradients estimate is computed by differentiating the updates of the learning algorithm and reversing its sequence of parameters (Lines 7-13), i.e., expanding the terms in Eq. (3.9) in reverse order. This approach can be derived by leveraging a Lagrangian formulation associated with the parameter optimisation dynamics (Franceschi et al., 2017). Lines 8-10 compute the corresponding Hessian-vector products, whereas Lines 11-12 update the value of the

Algorithm 3.2: Projected hypergradient descent/ascent for learning the hyperparameters and poisoning points

Input: $\mathcal{M}$, $\mathcal{A}$, $\mathcal{L}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{tr}}$, n_{p}, $\mathcal{P}$, T_{out}, T, γ, η
Output: $\mathcal{D}_{\text{p}}^{(T_{\text{out}})}$, $\boldsymbol{\Lambda}^{(T_{\text{out}})}$

$\mathcal{D}_{\text{p}}^{(0)} \leftarrow \texttt{initDp}\left(\mathcal{D}_{\text{tr}}, n_{\text{p}}\right)$ // Generate n_{p} initial samples for $\mathcal{D}_{\text{p}}^{(0)}$
$\mathcal{D}_{\text{tr}}^{\prime(0)} \leftarrow \left(\mathcal{D}_{\text{tr}} \setminus \{(\mathbf{x}_{\text{tr}_k}, y_{\text{tr}_k})\}_{k \in \mathcal{P}}\right) \cup \mathcal{D}_{\text{p}}^{(0)}$ // Replace n_{p} samples of $\mathcal{D}_{\text{tr}}$ by $\mathcal{D}_{\text{p}}^{(0)}$
$\boldsymbol{\Lambda}^{(0)} \leftarrow \texttt{initL}(\mathcal{M})$ // Initialise $\boldsymbol{\Lambda}$
for $\tau = 0$ **to** $T_{\text{out}} - 1$ **do**
  $\mathbf{w}^{(0)} \leftarrow \texttt{initW}(\mathcal{M})$ // Initialise $\mathbf{w}$
  $\nabla_{\mathbf{X}_{\text{p}}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right), \nabla_{\boldsymbol{\Lambda}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right) \leftarrow$
  $\texttt{hypGrad}\left(\mathcal{M}, \mathcal{A}, \mathcal{L}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{tr}}^{\prime(\tau)}, \boldsymbol{\Lambda}^{(\tau)}, \mathbf{w}^{(0)}, T, \eta\right)$ // RMD (Alg. 3.1)
  $\mathbf{X}_{\text{p}}^{(\tau+1)} \leftarrow \Pi_{\Phi(\mathcal{D}_{\text{p}})}\left(\mathbf{X}_{\text{p}}^{(\tau)} + \gamma \nabla_{\mathbf{X}_{\text{p}}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right)\right)$ // Projected Hypergradient Ascent
  $\mathbf{X}_{\text{tr}}^{\prime(\tau+1)} \leftarrow \left(\mathbf{X}_{\text{tr}}^{\prime(\tau)} \setminus \mathbf{X}_{\text{p}}^{(\tau)}\right) \cup \mathbf{X}_{\text{p}}^{(\tau+1)}$ // Update $\mathbf{X}_{\text{tr}}^{\prime(\tau)}$ with $\mathbf{X}_{\text{p}}^{(\tau+1)}$
  $\boldsymbol{\Lambda}^{(\tau+1)} \leftarrow \Pi_{\Phi(\boldsymbol{\Lambda})}\left(\boldsymbol{\Lambda}^{(\tau)} - \gamma \nabla_{\boldsymbol{\Lambda}} \mathcal{A}\left(\mathcal{D}_{\text{val}}, \mathbf{w}^{(T)}\right)\right)$ // Projected Hypergradient Descent
end for

hypergradients. We use a notation similar to Domke (2012); Maclaurin et al. (2015); Muñoz-González et al. (2017), where more details on the derivation of this algorithm can be found.

3.3.3 Projected Hypergradient Descent/Ascent

After computing the hypergradients, at each *hyperiteration* we use projected hypergradient descent/ascent to update the poisoning points and the hyperparameters:

$$\begin{aligned} \mathbf{X}_{\text{p}} &\leftarrow \Pi_{\Phi(\mathcal{D}_{\text{p}})}\left(\mathbf{X}_{\text{p}} + \gamma \, \nabla_{\mathbf{X}_{\text{p}}} \mathcal{A}\right), \\ \boldsymbol{\Lambda} &\leftarrow \Pi_{\Phi(\boldsymbol{\Lambda})}\left(\boldsymbol{\Lambda} - \gamma \, \nabla_{\boldsymbol{\Lambda}} \mathcal{A}\right), \end{aligned} \tag{3.10}$$

where γ is the learning rate for the outer problem and $\Pi_{\Phi(\mathcal{D}_{\text{p}})}$ and $\Pi_{\Phi(\boldsymbol{\Lambda})}$ are the projection operators for the features of the poisoning points, $\mathbf{X}_{\text{p}}$, and the hyperparameters, $\boldsymbol{\Lambda}$, defined as $\Pi_{\Phi(\,\cdot\,)}(\mathit{input}) \triangleq \text{clip}(\mathit{input}, \inf \Phi(\,\cdot\,), \sup \Phi(\,\cdot\,))$, so that their updated values are within the corresponding feasible domains, $\Phi(\,\cdot\,)$. In our case we used standard gradient descent/ascent to solve Eq. (3.3). The analysis of other alternatives to solve minimax games, such as (Daskalakis and Panageas, 2018), is left for future work.

Alg. 3.2 describes the procedure to solve the multiobjective bilevel problem proposed in

this chapter. Essentially, this algorithm implements projected hypergradient descent/ascent for T_{out} iterations (Lines 4-10) to optimise, in a coordinated manner, the poisoning points (Line 7)—replaced into the training set (Line 8)—and the set of hyperparameters (Line 9).

To reduce the computational burden, we consider the simultaneous optimisation of a batch of n_{p} poisoning points, $\mathcal{D}_{\text{p}} = \{(\mathbf{x}_{\text{p}_k}, y_{\text{p}_k})\}_{k=1}^{n_{\text{p}}}$. We generate the initial values of $\mathcal{D}_{\text{p}}$ by cloning n_{p} samples—uniformly sampled without duplicates—of $\mathcal{D}_{\text{tr}}$. Their labels are initially flipped and kept fixed during the optimisation. This process is carried out in the function `initDp` (Line 1). Then, these n_{p} poisoning samples replace the n_{p} clean samples of $\mathcal{D}_{\text{tr}}$ whose indices are in the set $\mathcal{P}$ (Line 2). On the other hand, the hyperparameters are initialised in `initL` (Line 3).

To solve the bilevel problem, every time the variables in the outer problem are updated, the model's parameters need to be previously initialised and optimised. Thus, let `initW` (Line 5) be a particular initialisation for the model's parameters. `hypGrad` (Line 6) refers to the particular optimisation algorithm used to train the model's parameters and compute the corresponding hypergradients. In this work, this algorithm is RMD (Alg. 3.1).

3.4 Regularisation to Partially Mitigate Poisoning Attacks

Poisoning attacks are intrinsically related to the stability of ML algorithms. Attackers aim to produce large changes in the target algorithm by influencing a reduced set of training points. Xu et al. (2011) introduced the following definition of stability: *"an ML algorithm is stable if its output is nearly identical on two datasets, differing on only one sample."* This concept of stability has also been studied in the field of robust statistics, in which "robustness" formally denotes this definition of stability (Rubinstein et al., 2009). It is not our intention here to provide a formal analysis of the stability of ML algorithms, but to show that stability is an important property in the design of ML algorithms that are robust to data poisoning.

L_2 (or Tikhonov) regularisation is a well-known mechanism to increase the stability of ML algorithms (Bousquet and Elisseeff, 2002; Xu et al., 2011). In L_2 regularisation, a penalty term is added to the original loss function, which shrinks the norm of the model's parameters, so that $\mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w}, \lambda) = \mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w}) + \frac{e^{\lambda}}{2}||\mathbf{w}_{1:(d-1)}||_2^2$, where λ is the hyperparameter that controls the strength of the regularisation term and $\mathbf{w}_{1:(d-1)}$ denotes

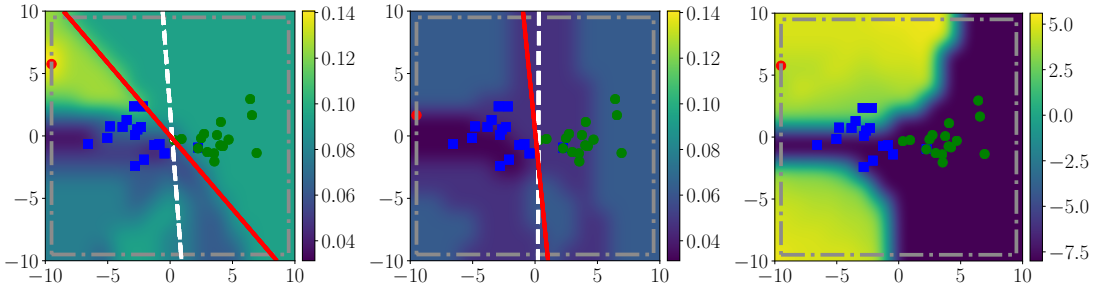


Figure 3.1: Effect of regularisation on a synthetic example. The blue and green points represent the training data points for each class, and the red point is the poisoning point (labelled as green). The dashed-dotted grey box represents the attacker's constraints. The dashed-white lines and solid-red lines depict the decision boundaries for LR classifiers trained on clean and poisoned datasets respectively. (a) Standard LR with no regularisation. (b) LR with L_2 regularisation. The colourmaps in the two plots represent the validation error as a function of the poisoning point. (c) Value of λ learnt by solving Eq. (3.1) as a function of the location of the injected poisoning point.

all the entries of $\mathbf{w}$ but the bias term.[6] The exponential form is used to ensure a positive contribution of the regularisation term to the loss function and to help learning λ, for example by using Eq. (3.1), as this hyperparameter is usually searched over a log-spaced grid (Pedregosa, 2016). In principle, different L_2 regularisation schemes can be considered: e.g., in neural networks, we could have a different regularisation term for each layer or even for each parameter (Foo et al., 2008).

Xiao et al. (2015) analysed the robustness of embedded feature selection, including L_2 and L_1 regularisation, for linear classifiers against optimal poisoning attacks. Although their experimental results showed that L_2 was slightly more robust compared to L_1 regularisation and *elastic-net*, all the classifiers tested where very vulnerable to indiscriminate optimal poisoning attacks. However, these results relied on the assumption that the regularisation hyperparameter was constant regardless of the fraction of poisoning data, which as we show in our experiments provides a limited perspective on the robustness of the learning algorithms.

The synthetic example with an LR classifier in Fig. 3.1 illustrates the limitations of the approach of Xiao et al. (2015). Here, the points were drawn from two different bivariate Gaussian distributions, $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ with parameters:

[6] Note that we do not regularise the bias term of the model, since it is a common practice in ML.

$$\boldsymbol{\mu}_0 = \begin{bmatrix} -3.0 \\ 0.0 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_0 = \begin{bmatrix} 2.5 & 0.0 \\ 0.0 & 1.5 \end{bmatrix},$$
$$\boldsymbol{\mu}_1 = \begin{bmatrix} 3.0 \\ 0.0 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 2.5 & 0.0 \\ 0.0 & 1.5 \end{bmatrix}.$$

The attacker uses 32 points (16 per class) for training and 64 (32 per class) for validation, and one poisoning point cloned from the validation set (in the example of Fig. 3.1, cloned from the set labelled as blue), whose label is flipped. This poisoning point is concatenated into the training set and the features of this point are optimised with RMD. In order to poison the LR classifier, we use $\gamma = 0.4$ and $T_{\mathcal{D}_\text{p}} = 50$; $\Phi(\mathcal{D}_\text{p}) \in [-9.5, 9.5]^2$; $\eta = 0.2$, $T = 100$; and when testing the attack, $\eta_\text{tr} = 0.2$, batch size = 32 (full batch), and number of epochs = 100. When we apply regularisation, we fix $\lambda = \log(20) \approx 3$.

Fig. 3.1(a) shows the effect of injecting a single poisoning point (red point, labelled as green) to maximise the error (measured on a separate validation set with 32 points per class) against a non-regularised LR classifier. The dashed-white line represents the decision boundary learnt when training on the clean dataset, and the red line depicts the decision boundary when training on the poisoned dataset. We observe that a single poisoning point can significantly alter the decision boundary. Fig. 3.1(b), shows a similar scenario, but training an LR classifier with L_2 regularisation, setting $\lambda = \log(20) \approx 3$. Here, we observe that the effect of the poisoning point is much reduced and the decision boundary shifts only slightly. In the background of these two figures we represent the validation error of the LR trained on a poisoned dataset as a function of the location of the poisoning point. We observe that, when there is no regularisation (Fig. 3.1(a)) the error can significantly increase when we inject the poisoning point in certain regions. On the contrary, when regularisation is applied (Fig. 3.1(b)), the colourmap is more uniform, i.e., the algorithm is quite stable regardless of the position of the poisoning point. Note that, when the model is regularised, the increase in the validation error after the attack is small. In Sect. 3.5, we also experiment with L_1 regularisation against data poisoning. In this case, $\mathcal{L}(\mathcal{D}_\text{tr}, \mathbf{w}, \lambda) = \mathcal{L}(\mathcal{D}_\text{tr}, \mathbf{w}) + e^\lambda ||\mathbf{w}_{1:(d-1)}||_1$.[6]

Fig. 3.1(c) shows how the optimal value of λ that minimises the loss in the trusted validation set changes significantly as a function of the location of the poisoning point. The colourmap in the background represents the value of λ. To plot the colourmap, the

Dataset	# Training Samples	# Validation Samples
MNIST ('0' vs. '8')	5,000	500
FMNIST (*trouser* vs. *pullover*)	5,000	500
CIFAR-10 (*airplane* vs. *frog*)	5,000	1,000

Dataset	# Test Samples	# Features
MNIST ('0' vs. '8')	3,000	784
FMNIST (*trouser* vs. *pullover*)	3,000	784
CIFAR-10 (*airplane* vs. *frog*)	2,500	3,072

Table 3.2: Characteristics of the datasets used in the experiments.

values of λ explored for each possible poisoning point are in the range $[-8, 6]$. Then, the optimal value of λ is chosen such that it minimises the error of the model, trained on each combination of the poisoning point (concatenated into the training set) and λ in the grid, and evaluated on the validation set.

We observe that λ is much bigger for the regions where the poisoning point can influence the classifier more (Fig. 3.1(c)). So, when the poisoning attack has a negative impact on the classifier's performance, the importance of the regularisation term, controlled by λ, increases. It is clear that selecting the value of λ appropriately, using a small trusted validation set, can have a significant impact on the classifier's robustness. Furthermore, when testing the robustness of regularised classifiers we must consider the interplay between the attack strength and the value of λ.

3.5 Experiments

We evaluate the effectiveness of the attack strategy in Eq. (3.3) against LR and feed-forward DNNs. We study the influence of L_2 and L_1 regularisation on the attack, providing an analysis of the robustness of the learning algorithms to worst-case scenarios for attacks with different strengths. Note that the analysis of optimal indiscriminate poisoning attacks against non-convex models is substantially more computationally difficult. Most previous work on optimal poisoning attacks focusses on linear classifiers and, to our knowledge, our study is the first to analyse the effect of regularisation against data poisoning on DNNs.

3.5.1 Experimental Settings

For both LR and DNNs, we use three different binary classification problems: MNIST ('0' vs. '8') (LeCun et al., 1998), FMNIST (*trouser* vs. *pullover*) (Xiao et al., 2017), and

Dataset (Model)	T_{out}	γ	$\Phi(\mathcal{D}_{\text{p}})$
MNIST ('0' vs. '8') (LR)	140	0.300	$[0.0, 1.0]^{784}$
FMNIST (*trouser* vs. *pullover*) (LR)	150	0.300	$[0.0, 1.0]^{784}$
CIFAR-10 (*airplane* vs. *frog*) (LR)	120	0.300	$[-1, 1]^{3,072}$
MNIST ('0' vs. '8') (DNN)	180	0.075	$[0.0, 1.0]^{784}$
FMNIST (*trouser* vs. *pullover*) (DNN)	150	0.100	$[0.0, 1.0]^{784}$
CIFAR-10 (*airplane* vs. *frog*) (DNN)	120	0.100	$[-1, 1]^{3,072}$

Dataset (Model)	$\Phi(\lambda)$	T	η
MNIST ('0' vs. '8') (LR)	$(-\infty, \log(5 \cdot 10^3)]$	140	0.10
FMNIST (*trouser* vs. *pullover*) (LR)	$(-\infty, \log(5 \cdot 10^3)]$	160	0.04
CIFAR-10 (*airplane* vs. *frog*) (LR)	$(-\infty, \log(10^5)]$	500	0.01
MNIST ('0' vs. '8') (DNN)	$(-\infty, \log(5 \cdot 10^3)]$	700	0.04
FMNIST (*trouser* vs. *pullover*) (DNN)	$(-\infty, \log(5 \cdot 10^3)]$	800	0.03
CIFAR-10 (*airplane* vs. *frog*) (DNN)	$(-\infty, \log(10^5)]$	800	0.03

Table 3.3: Experimental settings for the poisoning attack.

Dataset (Model)	η_{tr}	# Epochs
MNIST ('0' vs. '8') (LR)	0.10	140
FMNIST (*trouser* vs. *pullover*) (LR)	0.04	160
CIFAR-10 (*airplane* vs. *frog*) (LR)	0.01	500
MNIST ('0' vs. '8') (DNN)	0.04	700
FMNIST (*trouser* vs. *pullover*) (DNN)	0.03	800
CIFAR-10 (*airplane* vs. *frog*) (DNN)	0.03	800

Table 3.4: Experimental settings for testing the attacks.

CIFAR-10 (*airplane* vs. *frog*) (Krizhevsky et al., 2009). All datasets are balanced and drawn at random from the original joint pool of training and test points. The details for each dataset are included in Table 3.2.

All our results are the average of 10 repetitions with different random data splits for training, validation and test sets. Moreover, both MNIST and FMNIST sets are normalised to be in the range $[0, 1]^{784}$, whereas CIFAR-10 sets are normalised to be in the range $[-1, 1]^{3,072}$. For all the attacks, we measure the average test error for different attack strengths, where the number of poisoning points ranges from 0 (0%) to $1,750$ (35%). The size of the batch of poisoning points that are simultaneously optimised is 350 for all the datasets. For MNIST and FMNIST, this leads to $274,400$ features to be optimised simultaneously, and to $1,075,200$ features for CIFAR-10. In this way, we simulate six different ratios of poisoning ranging from 0% to 35%.

We simulate different ratios of poisoning points in a cumulative manner: Once the

optimisation of the current batch of poisoning points and hyperparameters is finished,[7] this batch of poisoning points is fixed and the next batch of poisoning points is replaced into the remaining clean training set, whereas the hyperparameters are re-initialised, to carry out their corresponding optimisation. To accelerate their optimisation, the hypergradients for the poisoning points are normalised with respect to their ℓ_2-norm, and the hypergradients for each $\mathbf{\Lambda}$ are also normalised with respect to their corresponding value.[8]

The LR classifier's parameters are always initialised with zeros, for all the datasets. The DNN models have two hidden layers with Leaky ReLU activation functions (negative slope $= 10^{-2}$) as follows: $784 \times 32 \times 8 \times 1$, i.e., $25,393$ parameters, for MNIST and FMNIST; and $3,072 \times 64 \times 32 \times 1$, i.e., $198,785$ parameters, for CIFAR-10. In the DNN models, these parameters are initially filled with values according to Xavier Initialisation method (Glorot and Bengio, 2010), using a uniform distribution for all the parameters except the bias terms, which are initialised with a value of 10^{-2}.

For all the experiments, we make use of SGD both to update the parameters in the forward pass of RMD, and to train the model when testing the attack (full batch training). The choice of the number of iterations for the inner problem, T, depends on the model and the training dataset. Low values of T could lead to low-quality approximations for the hypergradient. As T increases, the solution of RMD approaches the exact (true) hypergradient, but at the risk of overfitting the outer objective in the bilevel optimisation problem (Franceschi et al., 2018). The details of the attack settings are shown in Table 3.3, whereas the ones for testing the attacks are in Table 3.4.

All the experiments have been run on 2×11 GB NVIDIA GeForce® GTX 1080 Ti GPUs. The RAM memory is 64 GB (4×16 GB) Corsair VENGEANCE DDR4 3000 MHz. The processor (CPU) is Intel® Core™ i7 Quad Core Processor i7-7700k (4.2 GHz) 8 MB Cache.

[7] The criterion to finish the loop that optimises the variables of the outer level problem is given by the number of hyperiterations.

[8] The analysis of other techniques to accelerate the optimisation, such as adaptive learning rates, is left for future work.

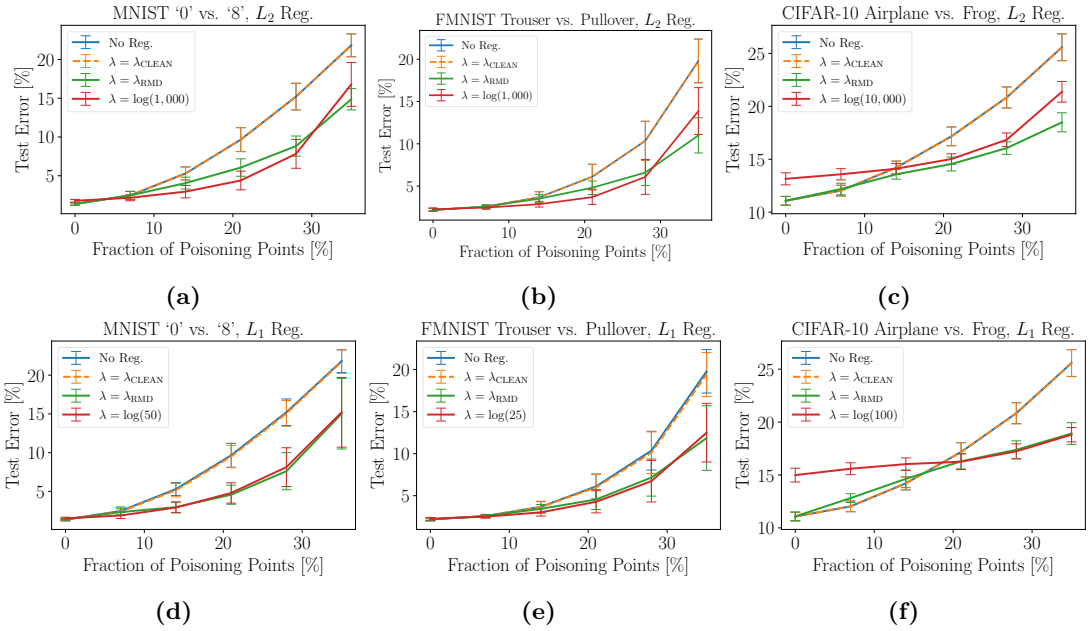
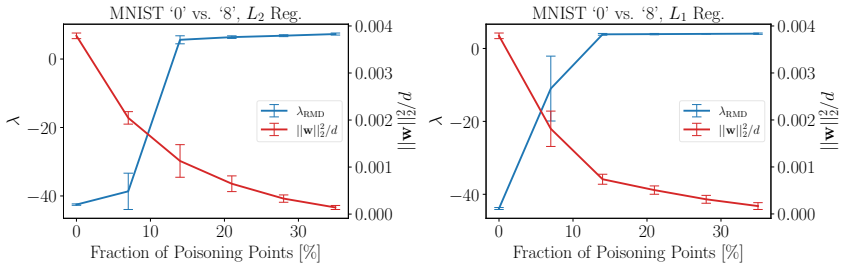


Figure 3.2: Average test error for the optimal attack against LR: The first row represents the case of L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. The second row contains the plots for L_1 regularisation on (d) MNIST, (e) FMNIST, and (f) CIFAR-10.

3.5.2 Logistic Regression

Test Error and Value of λ Learnt

For LR we test the general poisoning attack strategy in Eq. (3.3)—labelled as λ_{RMD} in the figures—using the following settings for the computation of the hypergradients with RMD. For MNIST we set T, the number of iterations for the inner problem, to 140. For FMNIST and CIFAR-10 we use $T = 160$ and $T = 500$, respectively. For comparison purposes, in addition to crafting attacks learning the value of λ, λ_{RMD}, we also craft optimal poisoning attacks setting the value of λ to different constant values: no regularisation ($\lambda = -\infty$); a very large one (for L_2 regularisation: $\lambda = \log(1,000)$ for MNIST and FMNIST, and $\lambda = \log(10,000)$ for CIFAR-10; for L_1 regularisation: $\lambda = \log(50)$ for MNIST, $\lambda = \log(25)$ for FMNIST, and $\lambda = \log(100)$ for CIFAR-10); and the value of λ optimised with 5-fold cross-validation (λ_{CLEAN}). By comparing with no regularisation and large constant values for λ, we aim to show the trade-off between accuracy (under clean data) and robustness to different attack strengths. The case of λ_{CLEAN} is similar to the settings used by Xiao et al. (2015), which uses a methodology akin to (Friedman et al., 2010), where the authors use K-fold cross-validation to select the value of λ, and the clean data is used both for training and validation in an unbiased way.



(a) **(b)**

Figure 3.3: Average λ and $||\mathbf{w}||_2^2$ for the optimal attack against LR on MNIST: (a) L_2 regularisation; (b) L_1 regularisation.

The results are shown in Fig. 3.2. We observe that when the model is not regularised or uses λ_{CLEAN}, the attacks are very effective and the test error increases significantly when compared to the algorithm's performance on the clean dataset (0% of poisoning). In contrast, for the largest λ the test error increases moderately with the increasing fraction of poisoning points, showing a lower test error compared to the case of no regularisation. However, in the absence of an attack, the algorithm *underfits* and the error is higher compared to the other models (especially in the case of CIFAR-10). When the value of λ is learnt (λ_{RMD}) using the trusted validation dataset, the increase in the test error is moderate and, when the ratio of poisoning points is large, the performance is similar to when λ is large. We can also observe that, in this case, when there is no attack, the performance is similar to that of the non-regularised classifier.

The results in Fig. 3.2 also show that the attack and the methodology presented by Xiao et al. (2015) provide an overly pessimistic view on the robustness of L_2 and L_1 regularisation to poisoning attacks, and that using the hyperparameter learnt when the data is clean can be detrimental under data poisoning. We show that, by appropriately selecting the value of λ, we can effectively reduce the impact of such attacks. We can also observe that there is a trade-off between accuracy and robustness: over-regularising (i.e., setting a very large value for λ) makes the algorithm more robust to the attack, but the performance on clean data is degraded.

In Fig. 3.3 and Fig. 3.4 we show the value of λ learnt and the norm of the model's parameters divided by the number of parameters, $||\mathbf{w}||_2^2/d$, as a function of the fraction of poisoning points injected. We observe that the regularisation hyperparameter increases and then saturates as we increase the fraction of poisoning points. Thus, the regularisation term compensates the effect of the poisoning points on the model's parameters up to a

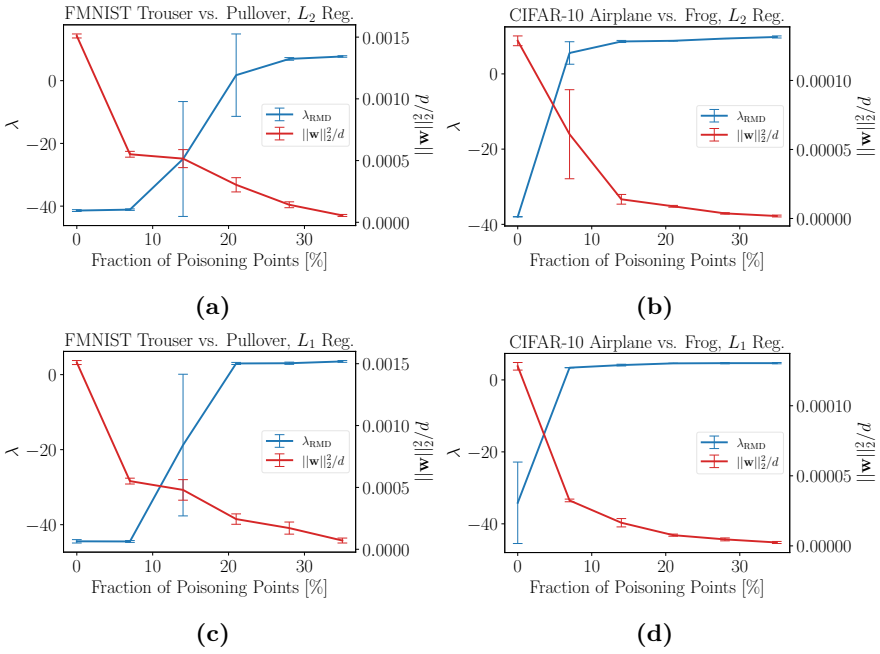


Figure 3.4: Average λ and average $||\mathbf{w}||_2^2$ for the optimal attack against LR: (a) L_2 regularisation on FMNIST, (b) L_2 regularisation on CIFAR-10, (c) L_1 regularisation on FMNIST, and (d) L_1 regularisation on CIFAR-10.

point.

Comparing L_2 and L_1, we observe that both regularisation techniques provide similar mitigation effects against the attack. Thus, even if L_1 regularisation does not necessarily provide stability to the learning algorithm, as is the case of L_2 regularisation, the use of the trusted validation set for learning the regularisation hyperparameter helps to mitigate the impact of the attack in both cases. The presence of the poisoning points increases the norm of the parameters if no regularisation is applied. But, when the trusted validation dataset is available for selecting the regularisation hyperparameter, both L_1 and L_2 regularisation can mitigate this effect, and thus, of reducing the impact of the poisoning points.

Sensitivity Analysis of the Size of the Validation Set

Next, we shall analyse the effect of the size of the trusted validation set. This analysis is important to understand its effect not only on the selection of the hyperparameters, but also on the effectiveness of the poisoning points learnt using the attack in Eq. (3.3) when evaluated on a separate test set. Note that having a larger trusted dataset is not necessarily beneficial only for the learner, but also for the attacker, who, under worst-case scenario assumptions, also has access to the trusted validation set.

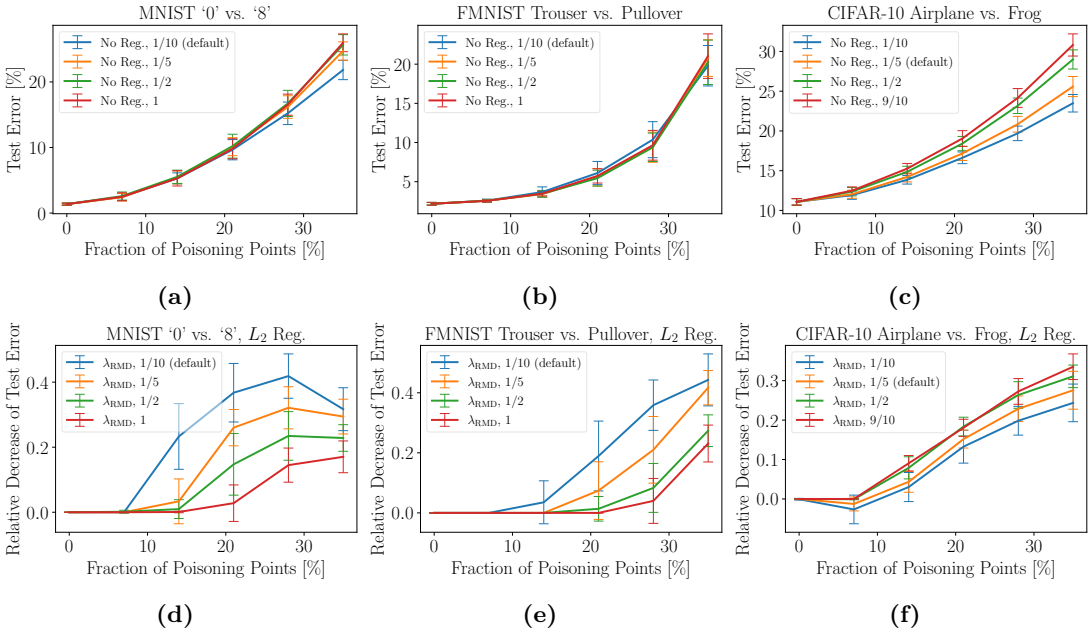


Figure 3.5: Sensitivity analysis of the size of the validation set. Results for (a) and (d) MNIST, (b) and (e) FMNIST, and (c) and (f) CIFAR-10. (a)—(c) Results when there is no regularisation. (b)—(f) Results for L_2 regularisation.

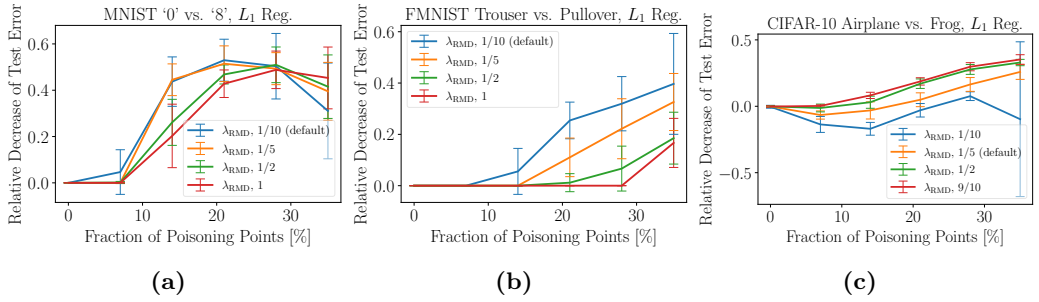


Figure 3.6: Sensitivity analysis of the size of the validation set for L_1 regularisation: (a) MNIST, (b) FMNIST, and (c) and CIFAR-10.

To study this effect, we consider an LR classifier and the same datasets (i.e., MNIST, FMNIST and CIFAR-10) and settings as before. Previously, we assumed that the validation set was ten times smaller than the training set for MNIST and FMNIST, and five times smaller for CIFAR-10. Now, the size of the training and test sets is fixed, and we evaluate different sizes for the validation set—compared to the size of the training set. To analyse the influence of the validation set both when there is no regularisation and when there is, we define the relative decrease of test error as the relative difference of the test error obtained when there is no regularisation and when the value of λ is learnt using the trusted validation set, i.e., $(\text{Test Error}_{|\text{No Reg.}} - \text{TestError}_{|\lambda_{\text{RMD}}})/\text{Test Error}_{|\text{No Reg.}}$.

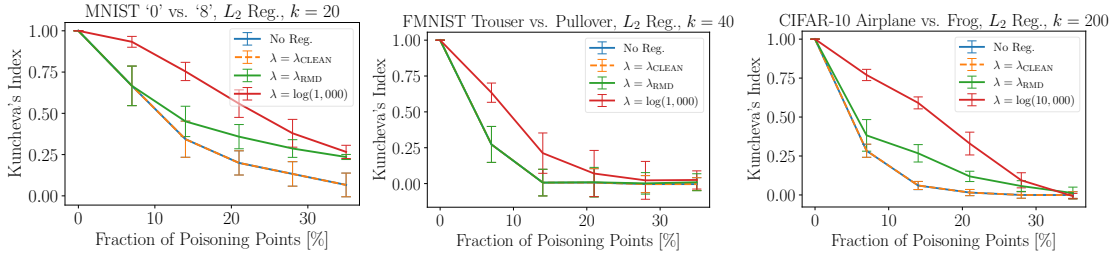
In Fig. 3.5 and Fig. 3.6, we observe that when the model is not regularised, for

MNIST and CIFAR-10, the test error is higher when the validation set is larger, as the poisoning points do not overfit the validation set. In contrast, for FMNIST the different-size validation sets result in a similar test error. On the other hand, when λ is learnt (L_2 and L_1 regularisation), for MNIST and FMNIST the test error decreases when the validation set is smaller, whereas for CIFAR-10, the opposite occurs. This shows that having a larger validation set is not always advantageous. When the poisoning points are learnt with no regularisation, a larger validation set provides more effectiveness for the attack, reducing the overfitting of the attack points. However, when using regularisation and the poisoning points and hyperparameters are jointly learnt, the optimal size of the validation set can be task-dependent. Our results show that, with this interplay between the learner and the attacker, the net benefit for the learner depends on the specific classification task, the size of the validation set and the attack strength. However, it is also important to note that, across all experiments, there is a clear benefit for using regularisation to mitigate the impact of the attack in all cases and, especially, for strong attacks.

Consistency Index

To understand how embedded feature selection methods based on L_2 regularisation are affected by the attack, we evaluate the stability of feature selection under poisoning using Kuncheva's consistency index (Kuncheva, 2007). Given two feature subsets $A, B \subseteq \mathcal{X}$, with $|A| = |B| = k, r = |A \cap B|$, and $0 < k < |\mathcal{X}| = d$, Kuncheva's consistency index is defined as $I_c(A, B) = (rd - k^2)/(k(d - k))$, where positive values indicate similar sets, zero is equivalent to random selections, and negative values indicate strong anti-correlation between the feature subsets. The underlying idea of this consistency index is to normalise the number of common features in the two sets using a correction for chance that accounts for the average number of common features randomly selected out of k trials (Xiao et al., 2015).

To evaluate how poisoning affects embedded feature selection, we compute this index using for A the feature set selected for the clean training data, and compare it against a set B selected under attack, at different percentages of poisoning. For each scenario, we consider the first k features exhibiting the highest absolute weight values: for MNIST, given that the most of the features are close to zero, we choose the top 20, 40 and 80 features; for FMNIST, the top 40, 80 and 160 features; and for CIFAR-10, the top 200,



(a) **(b)** **(c)**

Figure 3.7: Average Kuncheva's consistency index for the optimal attack against LR using L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10.

400 and 800 features.

The results for L_2 regularisation for MNIST, FMNIST and CIFAR-10 are shown in Fig. 3.7. The corresponding results for L_1 regularisation are consistent with these and can be found in Fig. 3.8. We observe that, in all cases, the consistency index decreases with the ratio of poisoning. This means that, to succeed, the attack naturally modifies the importance of the features of the training set (even if the attack is not specifically designed to do that), so that the poisoned model pays more attention to less relevant features. It is also clear that if the model is not regularised, the features selected are less consistent, and regularisation helps to increase the feature stability under poisoning. For λ_{RMD}, it is generally bounded between the cases of no regularisation and large value of λ, showing that the algorithm sacrifices some feature stability to decrease the test error. Compared to L_1 (Fig. 3.8), L_2 regularisation provides greater feature stability when using a large regularisation hyperparameter. It is important to note that the selection of the regularisation hyperparameter, using Eq. (3.3), aims to minimise the error on the validation set, not to maximise the stability of the features, which would require a different defensive strategy. However, the results in Fig. 3.7 help understand better the combined effect of the poisoning attack and the use of regularisation.

3.5.3 Deep Neural Networks

Poisoning attacks can have different effects on the different layers of the target DNNs. For instance, analysing the activations of the last hidden layer can be enough to detect some poisoning attacks. Intuitively, this makes sense because the early layers correspond to "low-level" features that are less likely to play an important role when learning the decision

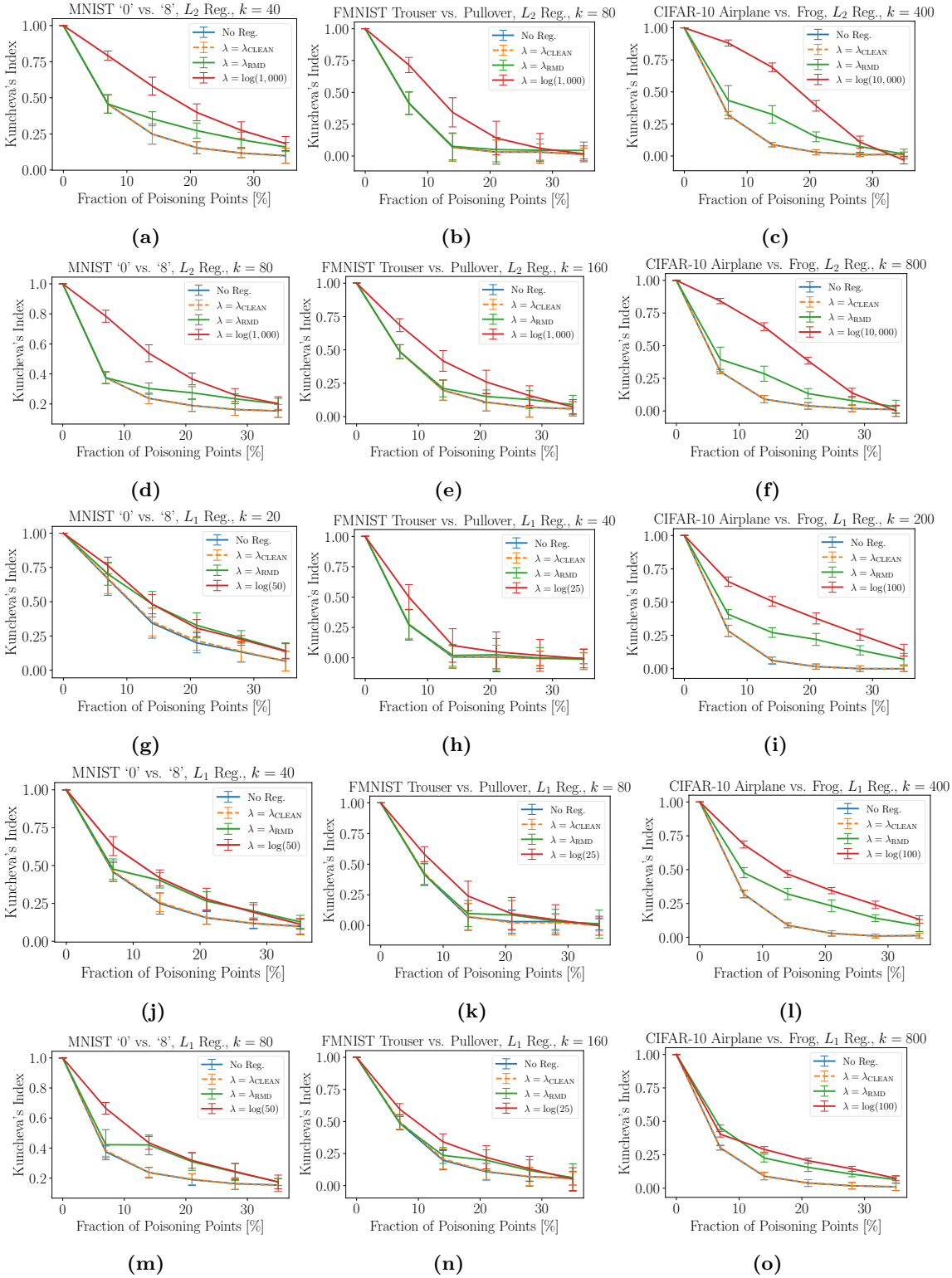


Figure 3.8: Average Kuncheva's consistency index for the optimal attack against LR, using L_2 regularisation (first and second rows); and using L_1 regularisation (third to fifth rows), on MNIST (first column), FMNIST (second column) and CIFAR-10 (third column), for different numbers of selected features, k: (a) $k = 40$, (b) $k = 80$, (c) $k = 400$, (d) $k = 80$, (e) $k = 160$, (f) $k = 800$, (g) $k = 20$, (h) $k = 40$, (i) $k = 200$, (j) $k = 40$, (k) $k = 80$, (l) $k = 400$, (m) $k = 80$, (n) $k = 160$, and (o) $k = 800$.

boundary (Chen et al., 2019). This problem has not been sufficiently studied in the research literature and, in this subsection, we provide useful insights that shed some light in this regard through the lens of regularisation. For this, we consider two possibilities: a single regularisation hyperparameter, and a vector of regularisation hyperparameters—with one hyperparameter for each layer. Intuitively, the amount of scaling needed by each layer's parameters to compensate for a change in the output is not the same, as the activation functions are non-linear. This also gives us an intuition about the most vulnerable layers to the poisoning attack. We also propose an additional modification to the RMD algorithm: we apply different initial random parameters $\mathbf{w}^{(0)}$ for every update of the poisoning points. This can be interpreted as assembling different randomly initialised DNNs to improve the generalisation of the poisoning points across different parameter initialisations. We set $T = 700$ for MNIST and $T = 800$ for FMNIST and CIFAR-10. This scenario is much more challenging for the bilevel problem we aim to solve, as the models have two hidden layers with Leaky ReLU activation functions (negative slope $= 10^{-2}$): $784 \times 32 \times 8 \times 1$, i.e., $25,393$ parameters, for MNIST and FMNIST; and $3,072 \times 64 \times 32 \times 1$, i.e., $198,785$ parameters, for CIFAR-10.

As before, we denote with λ_{RMD} the case where the regularisation hyperparameter is learnt according to Eq. (3.3), distinguishing now the cases: (i) when a single regularisation hyperparameter is used for the whole DNN, and (ii) when a different hyperparameter is used at each layer. We also performed attacks with different strength for the DNN assuming it is trained without regularisation ($\lambda = -\infty$) and with a large value for λ (for L_2 regularisation: $\lambda = \log(100)$ for MNIST, and $\lambda = \log(500)$ for FMNIST and CIFAR-10; for L_1 regularisation: $\lambda = \log(50)$ for MNIST, $\lambda = \log(10)$ for FMNIST, and $\lambda = \log(25)$ for CIFAR-10), constant for all the layers. Fig. 3.9 shows the results for L_2 regularisation. The results for L_1 regularisation are coherent with the ones for L_2 and can be found in Fig. 3.10. In this case, we omitted the case where λ is set with 5-fold cross-validation on the clean dataset as the search space is large, which makes it computationally very expensive.

The results in Fig. 3.9 are consistent with those obtained for the case of LR (Fig. 3.2). When there is no regularisation, the algorithm is vulnerable to the poisoning attack and its test error increases significantly. For a large value of λ, the algorithm's performance remains quite stable, but the clean error is higher. For λ_{RMD} the test error increases only moderately,

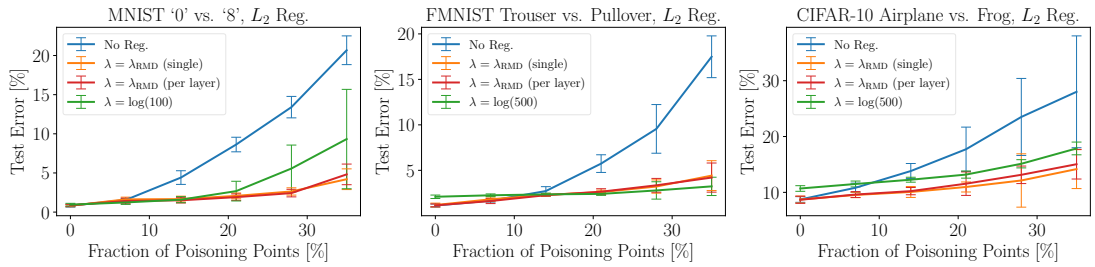


Figure 3.9: Average test error for the optimal attack against the DNNs using L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10.

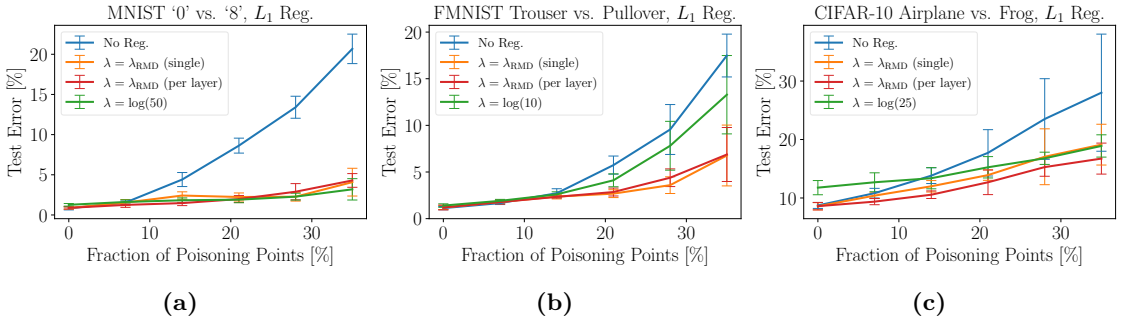
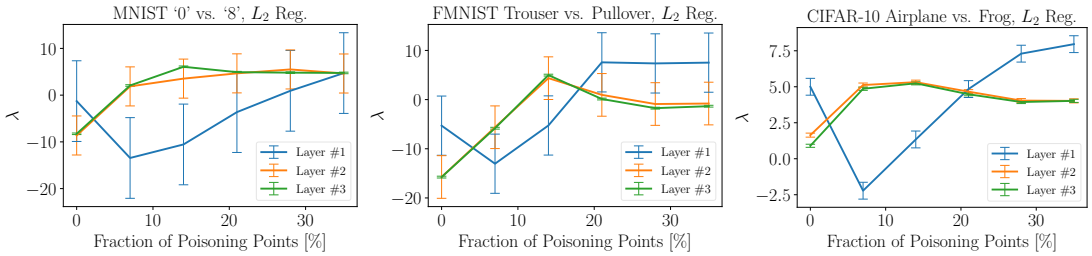


Figure 3.10: Average test error for the optimal attack against the DNNs using L_1 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10.

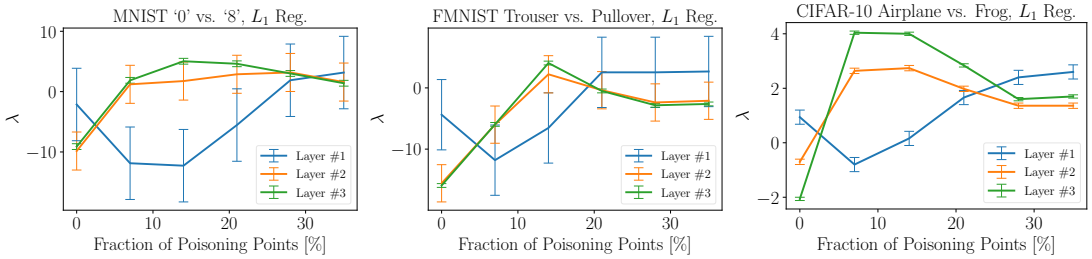
and the results when using a single hyperparameter or a different hyperparameter at each layer are very similar. From Fig. 3.9 and Fig. 3.10 we can see that when there is no attack, the test error for λ_{RMD} is smaller than in the other two cases. Although over-regularising may be appealing to make the algorithm more robust to poisoning, the performance in the absence of attacks may be significantly worse. Learning λ evidences this trade-off. For a large fraction of poisoning points, the small discrepancy observed between λ_{RMD} and the large value of λ is due to the non-convexity of the bilevel optimisation problem, resulting in learning (possibly) suboptimal values for λ_{RMD}. On the other hand, comparing the results for the DNNs (Fig. 3.9 and Fig. 3.10) and for LR (Fig. 3.2), it is evident that the mitigating effect of regularisation is more prominent in the case of DNNs. As the capacity of the DNN (compared to LR) is higher, the attackers can have more flexibility to manipulate the decision boundary. Hence, having regularisation in place, in combination with the trusted validation set, is even more important in the case of the DNNs.

Fig. 3.11 and Fig. 3.12 show the value of λ when using a different regularisation term at each layer, for L_2 and L_1 regularisation, correspondingly. We observe that the λ learnt for the second and output layers increases faster than the one for the first layer and, for



(a) **(b)** **(c)**

Figure 3.11: Average λ learnt with RMD at each layer of the DNNs, using L_2 regularisation, on (a) MNIST, (b) FMNIST, and (c) CIFAR-10.



(a) **(b)** **(c)**

Figure 3.12: Average λ learnt with RMD for the DNNs using L_1 regularisation with a different regularisation hyperparameter at each layer. The left, central and right figures represent the results for MNIST, FMNIST and CIFAR-10, respectively.

FMNIST and CIFAR-10, this increase is faster for the first hidden layer from 20% of poisoning. This suggests that the latter layers can be more vulnerable to the poisoning attack. The poisoning attack tries to produce more changes in those layers and, at the same time, the network tries to resist those changes by increasing the value of the corresponding regularisation hyperparameters. On the other hand, when the attack is very strong, the impact of the attack appears more uniform across all layers in the DNN, based on the values of λ learnt for each layer.

On the other hand, as in the case of LR, the value of the regularisation hyperparameters is also related to the norm of the weights divided by the number of parameters for each layer in the DNN. These results are shown in Fig. 3.13.

Finally, for the sake of completeness, Fig. 3.14 shows the value of λ learnt and the norm of the model's parameters divided by the number of parameters, $||\mathbf{w}||_2^2/d$, as a function of the fraction of poisoning points injected, for the DNNs when using a single regularisation term for L_2 and L_1 regularisation. These results are coherent with the ones for LR (Fig. 3.4 and Fig. 3.3). We observe that the regularisation hyperparameter

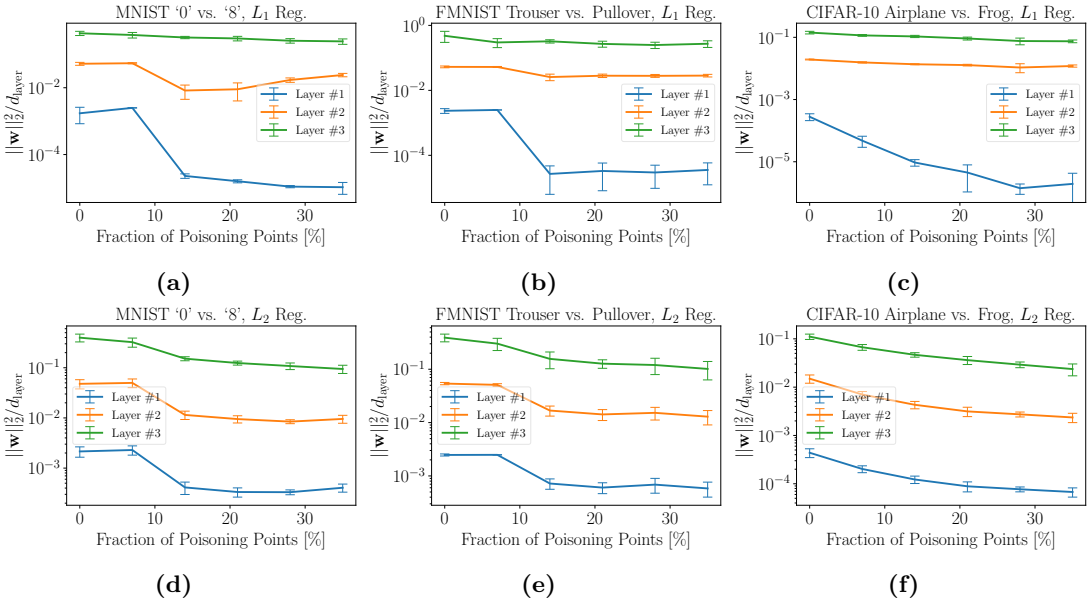


Figure 3.13: Average $||\mathbf{w}_{\text{layer}}||_2^2/d_{\text{layer}}$ for the DNNs using L_2 regularisation (first row) and L_1 regularisation (second row) with a different regularisation hyperparameter at each layer, where d_{layer} represents the number of parameters of the corresponding layer. This normalisation allows comparing $||\mathbf{w}_{\text{layer}}||_2^2$ for each layer regardless of the number of the parameters. The first, second and third columns represent the results for MNIST, FMNIST and CIFAR-10, respectively.

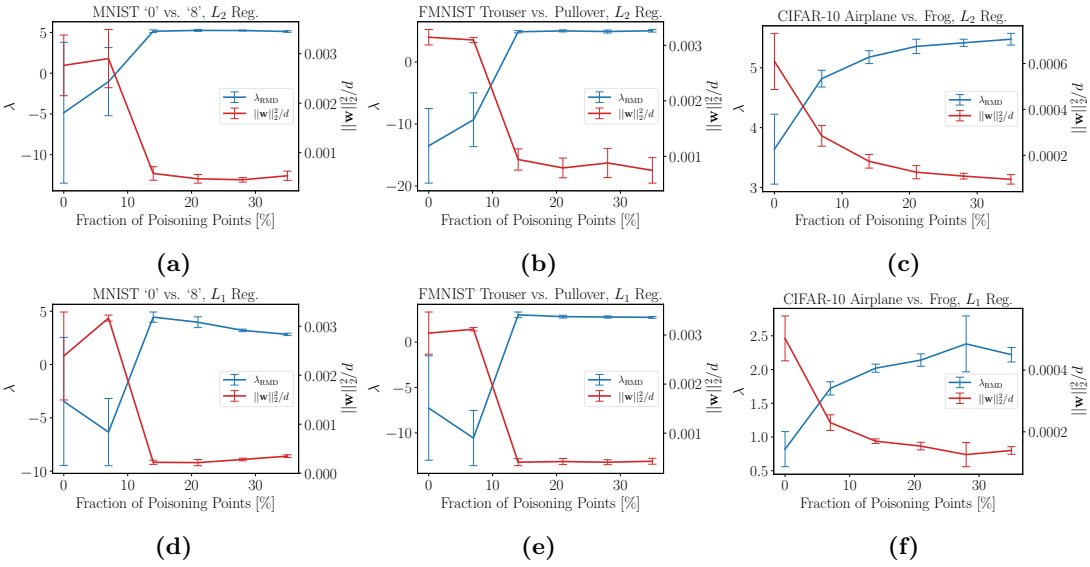


Figure 3.14: Average λ and average $||\mathbf{w}||_2^2/d$ for the optimal attack against the DNNs using a single regularisation hyperparameter, where d is the number of parameters of the model. The first row represents the case of L_2 regularisation on (a) MNIST, (b) FMNIST, and (c) CIFAR-10. The second row represents the case of L_1 regularisation on (d) MNIST, (e) FMNIST, and (f) CIFAR-10.

increases, and then, saturates as we increase the fraction of poisoning points. However, using a different regularisation at each layer can be more insightful to understand at which layer the attack focusses more.

3.6 Conclusions

Existing literature has been ambivalent on the role of regularisation in mitigating poisoning attacks. This problem has been insufficiently studied as existing works assume that regularisation hyperparameters are constant and chosen "a priori" regardless of the number of poisoning points or their effects.

For instance, Koh et al. (2022) reported some results on poisoning attacks against a linear SVM using L_2 regularisation. Their results suggest that, in some cases, increasing the value of the regularisation hyperparameter can make outlier detectors less effective. However, a direct comparison with our results is not possible as we consider a different threat model. Compared to the work of Koh et al. (2022), we have provided a more general formulation to model the effect of hyperparameters, including the case of L_2 regularisation, in the presence of data poisoning. Furthermore, we have provided a more complete and systematic evaluation of the benefits of using regularisation to mitigate the effect of poisoning attacks.

We have shown that the value of the hyperparameters depends on the amount of poisoning and that a constant value cannot be chosen a priori: when the value is too low, it provides insufficient robustness; when the value is too high, it damages performance. We have shown that when the value of the hyperparameters is learnt as a function of the poisoning incurred, regularisation can significantly mitigate the effect of indiscriminate poisoning attacks, whilst at the same time not damaging performance. This, however, requires the use of a small trusted validation set.

To study the mitigating effect of regularisation and choose hyperparameters, we have introduced a novel formulation where the poisoning attack strategy for worst-case scenarios is formulated as a *multiobjective bilevel optimisation problem*. This formulation allows to learn the most appropriate values for the model's hyperparameters and to calculate the poisoning points simultaneously. Solving this multiobjective bilevel optimisation problem is challenging. However, we have shown how this problem can be solved with gradient-based techniques by extending previous RMD-based approaches.

With this formulation, we have analysed the effect of indiscriminate poisoning attacks against LR and DNN classifiers when using both L_2 and L_1 regularisation. Our results confirm that the use of regularisation, combined with the presence of the small trusted

set to learn the hyperparameters, significantly helps to reduce the error under poisoning attacks. When the regularisation hyperparameter is learnt appropriately, the algorithm is more robust and, at the same time, the performance of the model is not affected when there is no attack. The trusted validation set required is quite small and task dependent; a larger trusted set is not necessarily advantageous.

Although L_2 regularisation typically provides more stability compared to L_1, our empirical results show that both types of regularisation are useful to reduce the effect of poisoning attacks. Additionally, our results show that the use of regularisation plays a more important role in more complex models, such as DNNs. Our empirical evaluation also shows that indiscriminate attacks have a more pronounced effect in the later layers of the network, as the value of the regularisation hyperparameters learnt for those layers increases significantly (with respect to those learnt when there is no attack) compared to the ones learnt for the first layers. However, for a large fraction of poisoning points, the effect of the attack is spread across all the different layers.

As for the road map of this thesis, in Ch. 4 we shall move on to analysing the vulnerabilities of regression models under data poisoning attacks, which have been somehow overlooked by the adversarial ML community. This will allow us to introduce a broader set of vulnerabilities that may arise in regression settings, compared to classification models. Moreover, in Ch. 5 we will introduce a novel stealth attack formulation built upon a multiobjective bilevel optimisation problem and solved through RMD, and a novel defence based on Bayesian linear regression. The closed-form solution of ridge regression will also allow us to analyse and introduce some fundamental theoretical properties of indiscriminate attack formulations in Ch. 6 and 7.

Chapter 4

Regression Poisoning: Threat Model

Synopsis

We propose a novel threat model for regression models. We present a large variety of error-specific attacks, including, for instance, assumptions violation, wrong model selection, interpretability violation, and uncertainty fluctuation. Moreover, we theoretically show that in linear regression models, any manipulation of a poisoning point directly affects all the parameters of the model. This result limits the possibility of creating very targeted attacks or backdoors in linear models. We also theoretically show that the feasible poisoning error increases with the "natural" noise level of the dataset and with the mismatch between the estimated and the true mapping function of the model. This result suggests that datasets that are difficult to fit to are also more difficult to defend when poisoned.

4.1 Introduction

So far, in this thesis we have analysed indiscriminate poisoning attacks in classification settings. However, regression algorithms are also critical in many applications. In particular, regression learning is applied in many safety-critical systems (for example, clinical and pharmaceutical industries (Paulson, 2006), control of cyber-physical systems (Akazaki, 2016), industrial processes for quality control (Sun and Hou, 2017), etc.), whilst the vast majority of the adversarial ML literature focusses on classification models. The

few studies that consider the robustness of regression models to poisoning attacks assume unrealistic threat models and are thus less useful in practice. Therefore, in the present chapter, we aim to evaluate several aspects of data poisoning attacks against regression learning, exceeding previous threat models both in terms of breadth and depth.

Motives for such attacks can be manifold: for example, personal motives (e.g., a malignant doctor or underpaid caregiver), financial motives (one company damaging another company's reputation, or an individual betting on the crash of some company's stock value), or even political motives (Müller et al., 2020).

Through the threat model proposed we show that poisoning attacks against regression models have a much broader range of attack objectives—compared to classification models—and thus a broader spectrum of attacks can be attempted. Combined with their widespread use, this makes regression models a prime and vulnerable target for poisoning attacks.

Hence, in this chapter we shall answer the following research question (Ch. 1):

- **RQ7**: What are the vulnerabilities of regression models under data poisoning attacks?

We shall first refer to the classical assumptions of regression models in stationary environments, and then discuss the possible violations that can occur in adversarial settings through our threat model. In the particular case of linear regression, we will theoretically show that any manipulation of a poisoning point directly affects all the parameters of the model. This result limits the possibility of creating very targeted attacks or backdoors in linear models. We will finalise the chapter by providing insights into an interesting property of regression models: the feasible poisoning error increases with the "natural" noise level of the dataset and with the mismatch between the estimated and the true mapping function of the model. This suggests that datasets that are difficult to fit to are also more difficult to defend when poisoned.

4.2 Regression Formulation and Classical Assumptions

Because we are studying regression models in offline environments, we assume that the order of the different data samples is not meaningful, i.e., there is no time component, and that the data is stationary—its distribution does not change over time. In these settings, researchers and practitioners often rely on a number of classical assumptions. These often include:

- The training samples are IID and representative of the population at large: there are not significant outliers.
- The features are measured with no noise (no *measurement errors*/*errors in variables*).
- The output noise is additive: there are not interactions with the inputs.
- All independent variables are uncorrelated with the output noise term.
- The mean of the output noise is zero (if it is not zero, it is immediate to correct the estimated bias term of the prediction).
- The distribution of the output noise (residuals) is typically Gaussian (at each value of the dependent variable).
- The expected correlation between residuals, for any two samples, is zero (uncorrelatedness assumption). Mathematically, the covariance matrix of the noise (across samples) is assumed to be diagonal.
- The variance of the residuals is constant across the input space (homoscedasticity). Its violation is called heteroscedasticity.
- No independent variables are a perfect linear combination of other independent variables, i.e., there are no high correlations amongst the independent variables (no perfect *multicollinearity*).

Practitioners have developed a variety of methods to maintain some or all of these desirable properties in real-world settings, because these classical assumptions are unlikely to hold exactly. For example, modelling *errors in variables* (*measurement errors*) (Durbin, 1954) can lead to reasonable estimates when the features are measured with errors.[1] Heteroscedasticity-consistent standard errors allow the variance of ϵ_i to change across values of $\mathbf{x}_i$. Correlated errors that exist within subsets of the data or follow specific patterns can be handled using *clustered standard errors* (Liang and Zeger, 1986), *geographic weighted regression* (Brunsdon et al., 1996), or *Newey-West standard errors* (Newey and West, 1987), amongst other techniques. When rows of data correspond to locations in space, the choice of how to model ϵ_i within geographic units can have important consequences.

[1]Typically, adding noise to the input features makes them less predictive of the response—in linear regression, it tends to push the parameters closer to zero than it would be if we regressed on the clean features.

Let us describe some of these assumptions more formally. We shall focus on single-output regression models (i.e., each input sample only has a single corresponding output). In the regression task, given the continuous input space $\mathcal{X} \subseteq \mathbb{R}^d$ and the continuous output space, $\mathcal{Y} \subseteq \mathbb{R}$, the learner aims to estimate the mapping $f : \mathcal{X} \to \mathcal{Y}$. Given a training set $\mathcal{D}_{\text{tr}} = (\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}) = \{(\mathbf{x}_{\text{tr}_i}, y_{\text{tr}_i})\}_{i=1}^{n_{\text{tr}}}$ with n_{tr} IID samples drawn from the underlying probability distribution $p(\mathcal{X}, \mathcal{Y})$, we can estimate f with a model $\mathcal{M}$ trained by minimising an objective function $\mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w})$ w.r.t. its parameters, $\mathbf{w} \in \mathbb{R}^d$ (if the model is parametric).

We assume the following additive model for the regression task:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, n, \tag{4.1}$$

where $\mathbf{x}_i$ is the vector of input features (a.k.a. independent, predictor, regressor, explanatory, measurement or covariate variables), y is the output label (a.k.a. dependent, predicted, regressand, explained or response variable), and the uncorrelated[2] noise ϵ_i (a.k.a. residual or error) is assumed to be homoscedastic, i.e., its power (variance) is constant and finite through the input space. This noise represents the irreducible *aleatoric uncertainty*[3] captured in the observations. This could be the case of, for example, sensor noise or motion noise, resulting in uncertainty which cannot be reduced even if more data were to be collected.

Given the additive model (4.1), a parametric hypothesis function, $f_{\mathbf{w}}(\mathbf{x})$, and a particular probability distribution for the zero-mean noise $\epsilon \sim p(\epsilon)$ (which is often assumed to be Gaussian), it is immediate to obtain the likelihood of the parameters $\mathbf{w}$ (or the observation $\mathbf{x}$ if the model is non-parametric) as $p(y|f_{\mathbf{w}}(\mathbf{x}))$. For example, assuming Gaussian noise, it is trivial to show that maximising this likelihood with respect to its parameters yields the same estimate of the parameters as does minimising the mean squared error (MSE). This justifies the use of the MSE as a maximum likelihood procedure, which—assuming the training data is noise-free in its features and sufficiently representative of the underlying data distribution—has several desirable statistical properties such as consistency and efficiency (for further details, please refer to statistics and statistical

[2]This uncorrelatedness (which does not necessarily imply independence) criterion assumes that the residuals of the output variables, y_i, are uncorrelated with each other, i.e., $\text{Cov}[\epsilon_i, \epsilon_j] = 0, \forall i \neq j$. For example, this is not necessarily the case for malicious perturbations in the output, as the poisoning samples may collude towards a particular goal.

[3]There exists another type of uncertainty, called *epistemic uncertainty*, which is subjective. It captures our ignorance about which model generated our collected data. This uncertainty can be explained away given enough data, and is often referred to as model uncertainty.

learning books, e.g., (Hastie et al., 2009)).

From an adversarial standpoint, it is important to notice that in existing regression methods, and discriminative methods in general, the input features $\mathbf{x}$ are typically assumed to be noise free. Due to this assumption, discriminative methods can experience a significant degradation in performance when out-of-distribution samples are present. The main difference with respect to classification settings is that the labels are continuous, which can provide different scenarios for the attacker. Since the labels are continuous, we do not consider human interpretation/bias when annotating the samples. This is because the output variable is the natural response of a particular system given its inputs. An attacker can however tamper with the inputs and/or output of the samples. So, in this case detectability constraints for the human perception are not very relevant (albeit they are relevant for the possible detection mechanism and the learning model). Let us have a look in more detail.

4.3 Threat Model

Poisoning attacks can be mainly characterised according to the attacker's *capability* to manipulate the training process, attacker's *goal*, and *knowledge* about the model and data.

4.3.1 Attacker's Capability

Typically, an attacker can tamper with a fraction of the training set to induce general or specific errors in the model learnt (Barreno et al., 2006, 2010). This is consistent with contexts in which the attacker has compromised part of the infrastructure, e.g., sensors from which the data is acquired or part of the data supply-chain, e.g., organisations from which the data is procured.

Here we describe the possible manipulations of the training data to craft the attacks. This is strongly related to the particular application domain. In these settings, we can distinguish three possible data manipulations for the attacker:

- Label manipulation attacks.
- Feature manipulation attacks.
- Data (both features and label) manipulation attacks.

We define $\mathcal{X}_{\text{source}}, \mathcal{X}_{\text{target}}, \mathcal{Y}_{\text{source}}$, and $\mathcal{Y}_{\text{target}}$ as the domains of the source features, target features, source labels, and target labels, respectively. We shall begin with the most general case, i.e., data manipulation attacks, where an attacker can tamper with both the features and labels of training data. We will define some general properties that can also be applicable for the other kind of manipulations. On the other hand, in the following subsubsections, we shall slightly abuse the notation and let $\boldsymbol{\theta}$ denote the target model's parameters and hyperparameters (here we will call them just parameters).

Data Manipulation Attacks

This is the most general attack. The attacker can manipulate both the features and labels of the training data to achieve a particular goal.

Definition 4.1 (Data manipulation attack)**.** *A data manipulation attack against a regression model is a continuous mapping* $[\boldsymbol{\tau}_x^T \quad \tau_y]^T : \mathcal{X}_{\text{source}} \times \mathcal{Y}_{\text{source}} \to \mathcal{X}_{\text{target}} \times \mathcal{Y}_{\text{target}}$*, where the source features* $\mathbf{x}_{\text{source}}$ *and source label* y_{source} *are modified as follows:*

$$\begin{bmatrix} \mathbf{x}_{\text{target}} \\ y_{\text{target}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}_x(\mathbf{x}_{\text{source}}, y_{\text{source}} | \mathcal{G}; \mathcal{D}_{\text{tr}}; \boldsymbol{\theta}) \\ \tau_y(\mathbf{x}_{\text{source}}, y_{\text{source}} | \mathcal{G}; \mathcal{D}_{\text{tr}}; \boldsymbol{\theta}) \end{bmatrix}, \tag{4.2}$$

where $\mathbf{x}_{\text{target}}$ *and* y_{target} *are the target features and label, respectively, and* $\mathcal{G}$ *represents the attacker's goals.*

The conditioning of the transformation functions on $\mathcal{G}$, $\mathcal{D}_{\text{tr}}$ and $\boldsymbol{\theta}$ denotes that these transformations *depend* on those variables. Depending upon the formulation of the attack, the transformations $\boldsymbol{\tau}_x$ and τ_y may or may not depend on the model's parameters $\boldsymbol{\theta}$. For example, in an optimal attack, the poisoning points are computed from the solution of the training problem, so that they implicitly encode the information of the model's parameters. In other cases, an attacker might fix an arbitrary transformation for the data regardless of the target model, to craft a transferable attack.

The family of transformation functions $[\boldsymbol{\tau}_x \ \tau_y]^\mathsf{T}$ considered by the attacker, which we call the *attacker's inductive bias*, is determined by the attacker's goal, budget, constraints and knowledge. For example, they can be defined by operations such as substitution by a constant, adding a constant, changing the sign, linear transformations, quadratic transformations, nearest neighbours, or even more complex functions. Also, depending on

the scenario, an attacker can be interested in a certain range of values of the inputs and outputs.

Label Manipulation Attacks

The attacker is only able to modify the labels of the data, which in this case are continuous.

Definition 4.2 (Label manipulation attack)**.** *A label manipulation attack against a regression model is a mapping* $\tau_y : \mathcal{Y}_{\text{source}} \to \mathcal{Y}_{\text{target}}$*, where the source label* y_{source} *is modified as follows:*

$$y_{\text{target}} = \tau_y(y_{\text{source}}|\mathbf{x}_{\text{source}}; \mathcal{G}; \mathcal{D}_{\text{tr}}; \boldsymbol{\theta}), \tag{4.3}$$

where y_{target} *is the target label.*

To avoid detection, the target-label domain is typically bounded. We can deduce that this attack generalises the label-flipping attack in classification settings, as in the latter the attack is a discrete mapping, whereas for regression, the attack is a continuous mapping. For example, an attacker can introduce a physical or digital transformation after the sensor measurements that could perturb the output. In this case, the attacker aims to transform a certain range of the sensor output into a different range of values.

Feature Manipulation Attacks

The attacker is only able to modify the features of the data. For instance, the attacker may want to control the maximum amount of perturbation added to the input data to have the attack points labelled as desired.

Definition 4.3 (Feature manipulation attack)**.** *A feature manipulation attack against a regression model is a continuous mapping* $\boldsymbol{\tau}_x : \mathcal{X}_{\text{source}} \to \mathcal{X}_{\text{target}}$*, where the source features* $\mathbf{x}_{\text{source}}$ *are modified as follows:*

$$\mathbf{x}_{\text{target}} = \boldsymbol{\tau}_x(\mathbf{x}_{\text{source}}|y_{\text{source}}; \mathcal{G}; \mathcal{D}_{\text{tr}}; \boldsymbol{\theta}). \tag{4.4}$$

Examples of this attack can be performed in situations where the attacker aims to modify a subset of features (e.g., a subset of the sensors that provide the input data), or to reduce correlations/importance of features to make the model ignore them. For instance, the manipulated regression coefficients can later be used by the adversary as a

backdoor of this learning system or mislead the interpretability of the regression model. In another example, changing the magnitude of a regression coefficient to small values can make its corresponding explanatory variable irrelevant. Similarly, the adversary can change the magnitude of a regression coefficient to a larger value to increase its importance. Furthermore, changing the sign of a regression coefficient can also lead us to misinterpret the correlation between its explanatory and response variables.

Attacker's Constraints

Another important aspect related to the attacker's capabilities is the possible presence of constraints for the manipulation of the data to craft the attacks. Often, the specific application domain can place constraints on the attacks that are possible. For example, in depth estimation applications, the attacker can possibly manipulate every pixel in the image with the only constrain of providing valid values for each pixel (Cheng et al., 2022). In another example in wireless sensor networks (Salem et al., 2014), an attacker may control only a subset of the sensors, and in some cases, events are triggered when measurements are higher or lower than a reference value, limiting the degrees of freedom to craft the attack.

We can characterise these constraints by assuming that an initial set of poisoning points is given and that the poisoning points are modified according to a space of possible modifications, as for example by constraining the norm of the perturbation of each poisoning point. This space of possible modifications can be included in the formulation of the attack. Note that detectability constraints in regression settings generalise the ones for classification, since the detectability constraints for the labels are also continuous.

4.3.2 Attacker's Goals

Both indiscriminate and targeted attacks can be error-generic or error-specific (see also Subsect. 2.3.2). In contrast to classification problems where error-specific poisoning attacks can target only a specific set of classes, in regression models the possible aims of an error-specific attack are much broader. For example, an attack may seek to cause specific variations of the predictions for a subset of samples that share certain characteristics, e.g., increasing by 10% the predicted price of properties within a garden in a specific area.

Table 4.1 summarises the possible attacks regarding the attacker's goal. However,

	Indiscriminate	Targeted
Error-Generic	Standard attacks aiming to increase the MSE on as many test samples as possible (our focus).	Attacks aiming to increase the MSE on a subset of samples that share a certain properties.
Error-Specific	Attacks aiming to make a particular kind of error on as many test samples as possible (e.g., modifying the bias term of the model to produce general overestimation/ underestimation in the model).	Attacks aiming to make a particular kind of error on a subset of samples that share certain properties (e.g., doubling the value of the labels whose original value is lower than one).

Table 4.1: Possible attacker's objectives depending on the specificity of the error and the targeted samples.

note that the range of error-specific attacks possible in regression models is limited by the complexity of the model. This is because in addition to inducing the desired error, error-specific attacks should not degrade the performance of the algorithm in other circumstances. For example, backdoor attacks (Liu et al., 2018b; Gu et al., 2019) seek to induce a specific behaviour when the trigger is present, but should not degrade the performance of the model when the trigger is absent. When the model is simple, e.g., linear regression, achieving such behaviour is difficult due to the simplicity of the model. Indeed, it is not possible to poison a linear regression model without modifying all of its parameters, as we shall show in Lemma 4.1. Intuitively, the different errors caused in the model are not independent of each other, and are more correlated in the case of linear models given their low expressive power. In particular, increasing different errors can make them also more detectable by looking at the MSE of the model.

Error Specificity

This very broad range of attack objectives makes attacks difficult to detect, difficult to defend against and regression problems highly vulnerable to such attacks. We can disambiguate cases where the nature of the errors can be different. Thus, according to the error specificity, an attack can be defined as:

- **Error-generic attack**: when the attacker aims to increase the overall error of the model. For example, when the attacker modifies an input so that the MSE of the regression model is increased in that input, regardless of the (wrong) predicted output.

To understand this goal, it can be illuminating to decompose the MSE between the true labels, Y, and the estimated predictions, $\hat{Y}$, as follows:

$$\text{MSE} = \mathbb{E}[(\hat{Y} - Y)^2] = \mathbb{E}^2[\hat{Y} - Y] + \text{Var}[\hat{Y} - Y]. \tag{4.5}$$

In this sense, the attacker's goal can be to tamper with (i) only the mean of the residuals—attacks aiming to produce general overestimation/underestimation in the model, by targeting the bias term in the model, see Fig. 4.1 as an example—(ii) the variance of the residuals; or (iii) both.

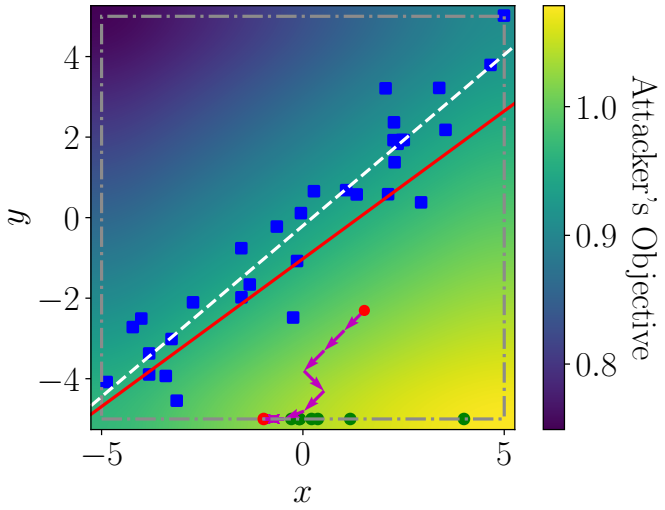


Figure 4.1: Synthetic example of an indiscriminate error-generic attack targeting the bias term of the model. The target model is linear regression. The blue points represent the clean training points, the green points represent the poisoning points, and the two red points represent a poisoning point at the beginning and at the end of its optimisation process, respectively, whose trajectory during the optimisation process is depicted by the magenta line. The colourmap represents the negative of the value of the attacker's objective, which aims to minimise the bias term of the model. The white dashed line is the regression line learnt under the clean data, the red line—which is translated w.r.t. the white line—represents the regression line learnt under the poisoning data, and grey bounding box represents the constraints of the poisoning points. Certainly, locating poisoning points in the yellow region can decrease the bias term of the model.

- **Error-specific attack**: when the attacker aims to produce a specific type of error. For example, an attacker can be interested in having certain predictions as close as a particular value as possible. Due to the variety of attacks that can occur in regression settings, we describe some possible attacks in Subsect. 4.3.3.

Samples Specificity

An attack may affect all the samples or be specific only to a subset of samples. We talk then of the *samples specificity*. Note that this is independent from the specificity of the error the poisoning seeks to induce, which may be error-generic or error-specific.

- **Indiscriminate attacks**: the adversary aims to degrade the performance of the system or to produce errors in an indiscriminate fashion, i.e., for a broad set of samples.

- **Targeted attacks**: the attacker aims to produce errors or to degrade the system performance for a reduced set of samples. Here the possible scenarios are much broader than in the case of classification. In this case, the source domains and/or the target domains are specified, and the task of the attacker is to craft the transformation that maps those domains. For example, $\mathcal{Y}_s$ can be restricted to a certain range of source labels.

On Targeted and Error-Specific Attacks

We shall discuss an important property of targeted and error-specific poisoning attacks against linear regression models.

Lemma 4.1. *In linear regression models, any manipulation of a poisoning point, either error-generic, error-specific, indiscriminate or targeted, directly affects all the parameters of the model, and therefore it directly affects the test MSE of the model:*

$$\text{MSE}_{\text{test}} = \frac{1}{n_{\text{test}}} ||\mathbf{X}_{\text{test}}\mathbf{w}^\star - \mathbf{y}_{\text{test}}||_2^2, \tag{4.6}$$

Proof. Let us find a relationship between the poisoned optimal model's parameters, $\mathbf{w}^\star$, and the clean optimal model's parameters, $\mathbf{w}^\star_{\text{tr}}$, for linear regression. Assuming a single poisoning point, $(\mathbf{x}_{\text{p}}, y_{\text{p}})$, let us define $\mathbf{X}_{\text{tr}_e} = [\mathbf{1}_{n_{\text{tr}}} \quad \mathbf{X}_{\text{tr}}]$ and $\mathbf{x}_{\text{p}_e} = \begin{bmatrix} 1 & \mathbf{x}_{\text{p}}^\mathsf{T} \end{bmatrix}^\mathsf{T}$. The closed-form solution to the least squares problem for linear regression, $\mathbf{w}^\star_{\text{tr}} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w})$, is $\mathbf{w}^\star_{\text{tr}} = \left(\mathbf{X}_{\text{tr}_e}^\mathsf{T}\mathbf{X}_{\text{tr}_e}\right)^{-1}\mathbf{X}_{\text{tr}_e}^\mathsf{T}\mathbf{y}_{\text{tr}}$. Substituting in the expression for $\mathbf{w}^\star$ when considering a single poisoning point, i.e., $\mathcal{D}_{\text{tr}} \cup (\mathbf{x}_{\text{p}}, y_{\text{p}})$, leads to:

$$\mathbf{w}^\star = \left(\mathbf{X}_{\text{tr}_e}^\mathsf{T}\mathbf{X}_{\text{tr}_e} + \mathbf{x}_{\text{p}_e}\mathbf{x}_{\text{p}_e}^\mathsf{T}\right)^{-1}\left(\mathbf{X}_{\text{tr}_e}^\mathsf{T}\mathbf{y}_{\text{tr}} + \mathbf{x}_{\text{p}_e}y_{\text{p}}\right). \tag{4.7}$$

Using the Sherman-Morrison formula (Appx. A.2):

$$\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e} + \mathbf{x}_{\mathrm{p}_e}\mathbf{x}_{\mathrm{p}_e}^\intercal\right)^{-1} = \left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1} - \frac{\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}\mathbf{x}_{\mathrm{p}_e}^\intercal\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}}{1 + \mathbf{x}_{\mathrm{p}_e}^\intercal\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}}. \tag{4.8}$$

And, substituting this inverse in the expression of $\mathbf{w}^\star$, we get:

$$\mathbf{w}^\star = \mathbf{w}_{\mathrm{tr}}^\star + \left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}\frac{y_{\mathrm{p}} - \mathbf{x}_{\mathrm{p}_e}^\intercal \mathbf{w}_{\mathrm{tr}}^\star}{1 + \mathbf{x}_{\mathrm{p}_e}^\intercal\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}}, \tag{4.9}$$

$$= \left(\mathbf{I}_d - \frac{\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}\mathbf{x}_{\mathrm{p}_e}^\intercal}{1 + \mathbf{x}_{\mathrm{p}_e}^\intercal\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}}\right)\mathbf{w}_{\mathrm{tr}}^\star + \frac{\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e} y_{\mathrm{p}}}{1 + \mathbf{x}_{\mathrm{p}_e}^\intercal\left(\mathbf{X}_{\mathrm{tr}_e}^\intercal \mathbf{X}_{\mathrm{tr}_e}\right)^{-1}\mathbf{x}_{\mathrm{p}_e}}, \tag{4.10}$$

which shows the strong dependence between the clean parameters, $\mathbf{w}_{\mathrm{tr}}^\star$, and the poisoned parameters, $\mathbf{w}^\star$, in linear regression. Therefore, any manipulation of a single feature or label of the poisoning point directly affects all the parameters of the original model. □

As a consequence, this result limits the possibility of creating backdoors in linear models, since one of the requirements of a backdoor attack is that the model should behave correctly under genuine samples.

4.3.3 Error-Specific Attacks

An attack against regression models can have a variety of different goals. Here we highlight some possible practical attacks, illustrated in the ML pipeline in Fig. 4.2.

Different attacks, e.g., that aim to maximise the energy consumption of the model, e.g., sponge poisoning (Cinà et al., 2022), are also possible in regression settings. These attacks are based on the observation that different inputs of the same size can cause a DNN (or other complex models) to draw very different amounts of time and energy. For instance, inputs that lead to less sparse models can increase the number of operations and the number of memory accesses, thus increasing the energy consumption.

Increase of Error

In this case, the attacker aims to increase the error of the prediction (e.g., in terms of increasing the value of the residuals for certain samples).

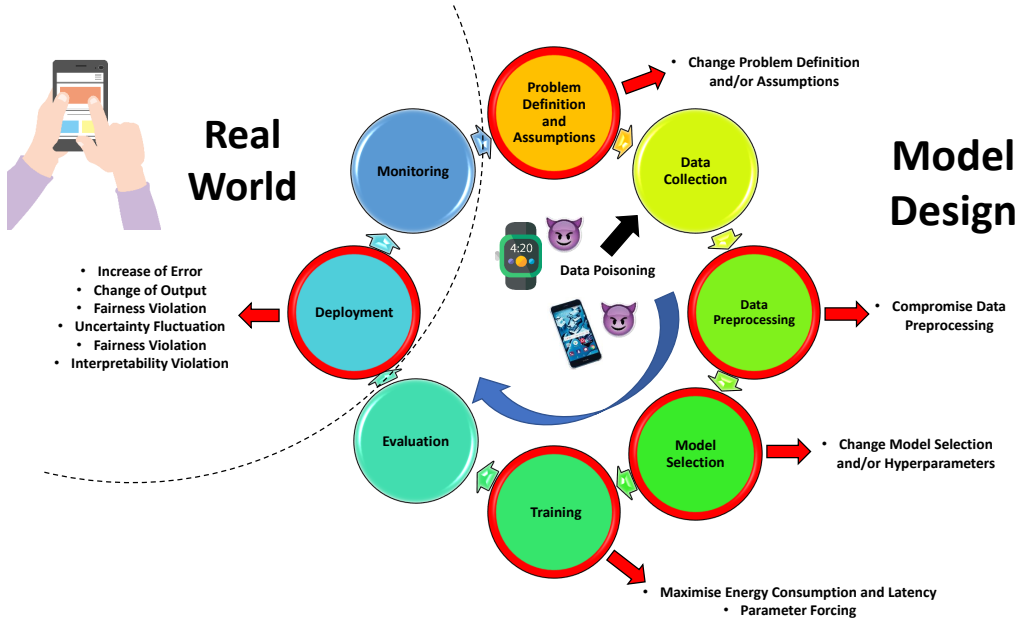


Figure 4.2: Possible error-specific attacks in regression settings. In the diagram, the black arrow denote the access point for the attacker in the ML pipeline. The red arrows denote the goals of the attacker. Each stage of the cycle can affect the next stages. Evaluation means that the model performance is checked before deployment. Deployment means "test" time in the real world, where we have previously unseen data. When monitoring the model, unexpected changes can lead to a redefinition of the problem and assumptions.

Change of Output

In some scenarios, the attacker can aim to force the model to predict specific output values for certain inputs. For instance, an attacker could force the output of a sensor to have a particular value, e.g., to saturate it over its dynamic range.

Assumptions Violation

The attacker aims to change the assumptions of the model, by changing the properties of the underlying data distribution (e.g., transforming the homoscedastic noise assumption into a heteroscedastic one; or in general breaking the stationarity assumption, e.g., changing the trend of the output for certain ranges of inputs). As an example, see Fig. 4.3.

Data Preprocessing

One of the attack vectors that have been overlooked in the research literature is data preprocessing. In this regard, a very common practice to achieve good performance in

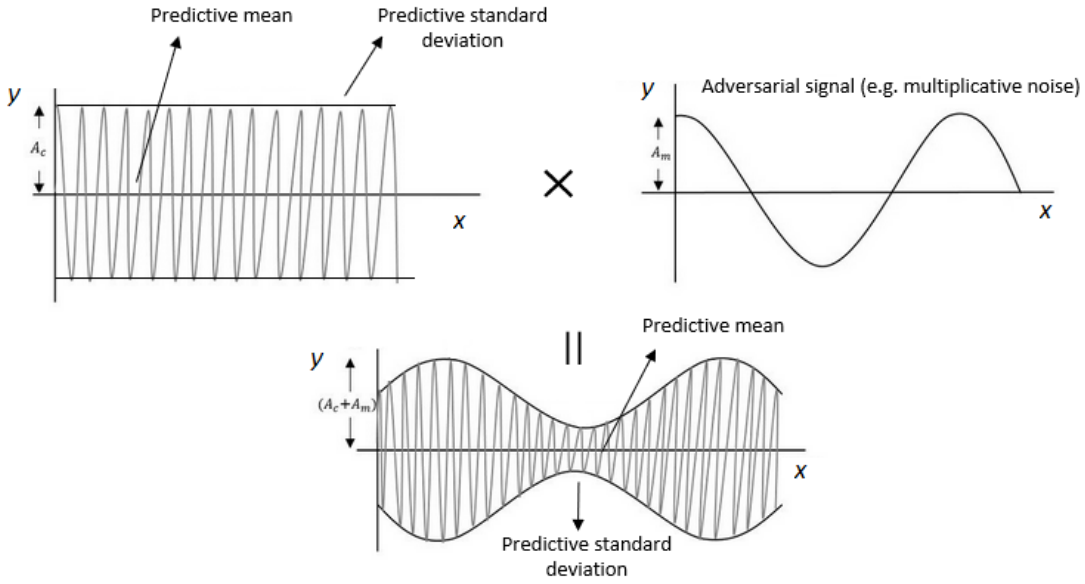


Figure 4.3: Example of a possible heteroscedastic attack inspired by the concept of amplitude modulation in signal processing. In this example, the genuine data ("carrier") is multiplied by an adversarial noise that varies through the input space. The result is the poisoned data, whose mean is similar and whose residual variance changes through the input space. For instance, this form of noise can lead to new forms of detectability constraints for the attacker.

the design of ML models is the normalisation of the features. Thus, in the case of data poisoning, when an attacker compromises the data collection, they already compromise the data preprocessing. However, little is known about how to adapt attacks when the defender transforms data prior to model estimation (Alfeld et al., 2019). For example, if the features are normalised using the data in the training set, when poisoning, the statistical properties of the training dataset are changed as well. Thus, the mean and the standard deviation for the features in the poisoned training dataset are different from the statistical properties of the features in the test set, and therefore, the performance of the system will be affected.

Wrong Model Selection

The attacker aims to make the learner select/obtain an incorrect model (inductive bias) and/or hyperparameters to overfit or underfit to the data. For example, in polynomial regression, the attacker can be interested in forcing or changing the polynomial degree (hyperparameter).

Interpretability Violation

The attacker aims to change the interpretability of the model, i.e., how we interpret the predictions of the model using its internal representations and parameters (Li et al., 2021). That is, the ability to interpret or to present a model in understandable terms to a human.

In this sense, the attacker can aim to manipulate the sensitivity of specific features in the model. In other words, attacks aiming at manipulating some of the parameters of the model. These attacks can have an effect of the interpretability of the model, as it is possible to eliminate or emphasise the effect of some of the features in the model. For example, to estimate the dose for a certain drug, an attacker could eliminate the blood pressure as a relevant factor for estimating the dose. Or, on the contrary, they could create untrue correlations between certain features and the target.

Fairness Violation

Algorithmic fairness deals with the design of algorithms that are not discriminatory, i.e., that do not introduce an unjustified disadvantage for members of a group, and particularly that do not further place at a disadvantage members of an already disadvantaged social group. In the case of regression, these "groups" can be denoted by categorical features.

The purpose of an attack can be to create or increase a disadvantage against a specific group of individuals or samples. For that, an attacker can add algorithmic discrimination, or exaggerating it if it already exists. In other words, the purpose of the attacker is to create or increase a disadvantage against a specific group of individuals or samples (Solans et al., 2020; Mehrabi et al., 2021; Van et al., 2022).

For instance, the developer of a house pricing prediction tool could sample training data in a discriminatory manner to bias the tool against a certain group of people. Similar harms can occur when training data is collected from public sources, such as online surveys that cannot be fully trusted. A minority of ill-intentioned users could poison this data to introduce defects in the ML system created from it. In addition to these examples, there is the unintentional setting, where inequities are introduced in the ML model as an undesired effect of the data collection. For instance, data of people corresponding to people of a certain ethnicity could be undersampled.

Mathematically, in order to change the influence of different sets, it is possible to

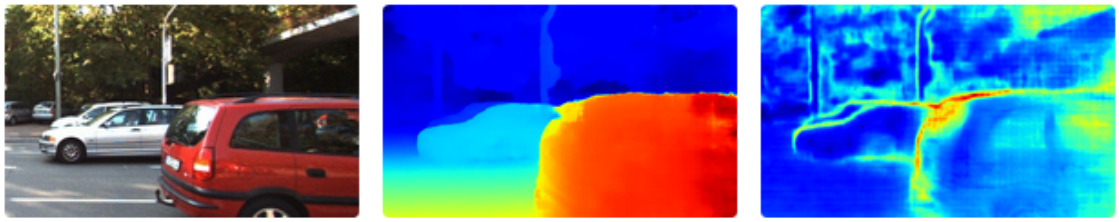


Figure 4.4: Uncertainty in depth estimation. The first image is an example input into a Bayesian model which estimates depth, as shown by the second image. The third image shows the estimated uncertainty. We can observe that the model predicts the wrong depth on difficult surfaces, such as the red car's reflective and transparent windows. Thankfully, the Bayesian model is also aware it is wrong and exhibits increased uncertainty (Kendall, 2017).

impose or weight special loss functions or penalty terms corresponding to different groups of people with their expected outcome (Solans et al., 2020; Mehrabi et al., 2021; Van et al., 2022).

Parameter Forcing

The attacker aims to force the model to learn a specific set of parameters—this is similar to the attacks proposed by Mei and Zhu (2015). This can include cases where the attacker just considers a subset of the parameters.

Uncertainty Fluctuation

There is no agreed definition of uncertainty in statistics. Some attempts to define it are: "incomplete knowledge in relation to a specified objective" (Bernardo and Smith, 2009) or "lack of knowledge regarding an unknown quantity" (Frey and Rhodes, 1996). In particular, the two most important types of uncertainty are aleatoric (i.e., stochastic) and epistemic (i.e., subjective). Aleatoric uncertainty refers to the notion of randomness, that is, the variability in the outcome of an experiment which is due to inherently random effects. This type of uncertainty is not a property of the model, but rather is an inherent property of the data distribution, and hence, it is irreducible. In contrast, epistemic uncertainty (a.k.a. knowledge uncertainty) occurs due to inadequate knowledge (about the best model). As opposed to uncertainty caused by randomness, uncertainty caused by ignorance can in principle be reduced on the basis of additional information. (Hüllermeier and Waegeman, 2021; Abdar et al., 2021). Aleatoric uncertainty can further be categorised into homoscedastic uncertainty, uncertainty which stays constant for different inputs, and

heteroscedastic uncertainty. Heteroscedastic uncertainty depends on the inputs to the model, with some inputs potentially having more noisy outputs than others (Kendall and Gal, 2017).

In Bayesian parametric models, epistemic uncertainty is modelled by placing a prior distribution over a model's parameters, and then trying to capture how much these parameters vary given some data. In contrast, aleatoric uncertainty is modelled by placing a distribution over the output of the model. For instance, in regression the outputs might be modelled as corrupted with Gaussian random noise (Kendall and Gal, 2017).

In adversarial settings, the attacker can hence target the uncertainty of the predictions. In terms of aleatoric uncertainty, we can distinguish two kind of attacks: *homoscedastic* and *heteroscedastic* attacks. Intuitively, in the former case the goal of the attacker is to increase or decrease the value of the noise variance constantly across the input the space; whereas in the latter, the goal of the attacker is to vary the noise variance through the input space. With respect to epistemic uncertainty, the attacker can aim to increase the variance of the parameters, or force the learner to change the prior distribution over the parameters.

As a practical example, heteroscedastic uncertainty is especially important for computer vision applications. For instance, for depth estimation, highly textured input images with strong vanishing lines are expected to result in confident predictions, whereas an input image of a featureless wall is expected to have very high uncertainty (see Fig. 4.4). Other situations that can provide uncertainty can be occlusion or specular reflection (Huang et al., 2015; Kendall, 2017).

4.3.4 Attacker's Knowledge

The attacker's knowledge comprises its knowledge about the learning model, about the data and about the defences employed to mitigate the attack. These can vary from ZK or LK attacks to PK attacks where the attacker is supposed to know *everything*. This creates a spectrum of possible situations with most realistic situations lying somewhere along this spectrum. For a more complete description of the attacker's knowledge, see Subsect. 2.3.2.

After introducing our novel threat model in regression settings, we will finalise this chapter analysing an interesting property: the difficulty of poisoning a regression model

trained on a particular dataset depends on the characteristics of the dataset, particularly on the "natural" noise of the dataset.

4.4 On the Difficulty of Poisoning Regression Models

Most of the attacks considered so far in the literature (Liu et al., 2017; Jagielski et al., 2018; Müller et al., 2020; Wen et al., 2021) generally assume that the sole goal of the attacker is to maximise the effect of the attack. Thus they are formulated as an optimisation problem where the attacker seeks to find perturbations that produce the maximum effect. This often induces perturbations that are more significant and leads to poisoning points that are more easily detectable (Jagielski et al., 2018) and thus easier to defend against.

More importantly, this is unrealistic. A defender detecting such perturbations may discard the training set entirely and forensically review the data acquisition chain that led to it. This would not only thwart the attack, but also undo the attacker's efforts to compromise the data supply-chain. It may also lead to attribution and further consequences. Thus, a data poisoning attack on an ML algorithm *must consider the risk of it being detected* and seek to minimise detectability risks. This has not been considered in the data poisoning research on regression algorithms so far.

Before considering attacks that also take into account detectability, which we will address in Ch. 5, it is helpful to understand the difficulty of poisoning regression models in such settings.

Definition 4.4 (Feasible poisoning error)**.** *We define the feasible poisoning error as the error induced in a particular model on a particular poisoned dataset for a fixed ratio of poisoning, when the poisoning points remain undetected by the defender.*

In the case of regression, the feasible poisoning error can be expressed as $\mathbb{E}[(Y - \hat{Y})^2]$, s.t. $\hat{Y} = \hat{f}(X)$, where $\hat{f}$ is learnt on the poisoned training set where the poisoning points are constrained by the defence mechanism.

Theorem 4.1. *The feasible poisoning error increases with the noise level ϵ and is given by:*

$$\mathbb{E}[(Y-\hat{Y})^2] = \text{Bias}^2[\hat{f}] + \text{Var}[\hat{f}] + \text{Var}[\epsilon] + 2\mathbb{E}[(\epsilon + \epsilon_\text{p})(f - \hat{f})] + 2\text{Cov}[\epsilon, \epsilon_\text{p}] + \mathbb{E}[\epsilon_\text{p}^2]. \quad (4.11)$$

Proof. Let us assume first the best possible oracle detector which *knows* the true input distribution $\mathcal{X}$ and the true mapping function $f(\mathcal{X})$. Then, the relationship between the output $\mathcal{Y}$ and $\mathcal{X}$ can be written in the general form as $Y = f(X) + \epsilon$, where ϵ is the zero-mean random error term, which is independent of X. If this detector rejects points whose label value is outside the range $f \pm \gamma$, where γ is lower bounded by ϵ, then it is straightforward to see that the error caused by any sample increases with ϵ. The reason is that the regions where the poisoning points can be injected without being detected are larger. Thus, under ideal situations where the random error tends to 0, it is more difficult to poison the model.

More formally, we now consider an attacker able to poison the labels with a poisoning perturbation ϵ_p, through the following generative process:

$$Y = f + \epsilon + \epsilon_\mathrm{p}. \tag{4.12}$$

In the general case, f is not known, but estimated through a function $\hat{Y} = \hat{f}(X)$ where $\hat{f}$ is not a perfect estimate for f, and this inaccuracy introduces some error. The poisoning perturbation ϵ_p is constrained to remain undetected, i.e., $\hat{f} + \epsilon_\mathrm{p} < \gamma$, where the detector now rejects samples outside the range $\hat{f} \pm \gamma$. Assuming that the poisoning perturbation follows a certain distribution and that it is not necessarily independent of X and $\hat{f}$—in most cases it will be dependent—the feasible poisoning error is given by:

$$\begin{aligned}
\mathbb{E}[(Y - \hat{Y})^2] &= \mathbb{E}[(f + \epsilon + \epsilon_\mathrm{p} - \hat{f})^2] \\
&= \mathbb{E}[(f - \hat{f})^2] + 2\mathbb{E}[(\epsilon + \epsilon_\mathrm{p})(f - \hat{f})] + \mathbb{E}[(\epsilon + \epsilon_\mathrm{p})^2] \\
&= \mathbb{E}[(f - \mathbb{E}[\hat{f}] + \mathbb{E}[\hat{f}] - \hat{f})^2] + 2\mathbb{E}[(\epsilon + \epsilon_\mathrm{p})(f - \hat{f})] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[\epsilon\epsilon_\mathrm{p}] + \mathbb{E}[\epsilon_\mathrm{p}^2] \\
&= \mathbb{E}[(f - \mathbb{E}[\hat{f}])^2] + 2\mathbb{E}[(f - \mathbb{E}[\hat{f}])(\mathbb{E}[\hat{f}] - \hat{f})] + \mathbb{E}[(\mathbb{E}[\hat{f}] - \hat{f})^2] \\
&\quad + \mathrm{Var}[\epsilon] + 2\mathbb{E}[(\epsilon + \epsilon_\mathrm{p})(f - \hat{f})] + 2\mathrm{Cov}[\epsilon, \epsilon_\mathrm{p}] + \mathbb{E}[\epsilon_\mathrm{p}^2] \\
&= \underbrace{\mathrm{Bias}^2[\hat{f}] + \mathrm{Var}[\hat{f}] + \mathrm{Var}[\epsilon]}_{\text{clean error}} + \underbrace{2\mathbb{E}[(\epsilon + \epsilon_\mathrm{p})(f - \hat{f})] + 2\mathrm{Cov}[\epsilon, \epsilon_\mathrm{p}] + \mathbb{E}[\epsilon_\mathrm{p}^2]}_{\text{poisoning error}},
\end{aligned}$$

where $\mathrm{Bias}[\hat{f}] = \mathbb{E}[\hat{f}] - f$. □

Therefore, the feasible poisoning error increases with the zero-mean noise level ϵ, which is an irreducible error component that all models incur, due to the intrinsic variability of the generative process. This result also suggests that a reasonable strategy for the attacker

to evade defences is to emphasise the "natural" noise of the dataset. We will empirically test this hypothesis in Ch. 5. On the other hand, it is also evident that the poisoning error is amplified by the mismatch between f and $\hat{f}$, which can be reducible by both using the most appropriate learning model to estimate f and also collecting more samples that are representative of the underlying data distribution.

4.5 Conclusions

In this chapter we have introduced a threat model for poisoning attacks against regression models. We started by stating the classical assumptions for regression settings, e.g., IID, no errors in variables, additive output noise, independent variables uncorrelated with the output noise, homoscedasticity, etc., and we highlighted that these assumptions can be violated in adversarial settings. From an adversarial standpoint, it is noteworthy that in existing regression methods, and discriminative methods in general, the input features are typically assumed to be noise free. Due to this assumption, discriminative methods can experience significant degrades in performance when out-of-distribution samples are present.

In terms of the threat model, attacks against regression algorithms generalise the ones against classification (in terms of the attacker's objective), since the labels in the former are continuous. we have presented a large variety of error-specific attacks, including, for instance, assumptions violation, wrong model selection, interpretability violation, and uncertainty fluctuation.

We also have shown that in linear regression models, any manipulation of a poisoning point, either error-generic, error-specific, indiscriminate or targeted, directly affects all the parameters of the model, and therefore it directly affects the test MSE of the model. This result limits the possibility of creating very targeted attacks or backdoors in linear models, since one of the requirements of a backdoor attack is that the model should behave correctly under genuine samples.

We have also theoretically shown that the feasible poisoning error increases with the "natural" noise level of the dataset and with the mismatch between the estimated and the true mapping function of the model. This result suggests that datasets that are difficult to fit to by a particular model are also more difficult to defend when poisoned.

Once we have analysed the possible attacks that can arise in regression settings, in Ch. 5 we shall focus on indiscriminate attacks against regression models. This will allow us to introduce novel attack formulations with detectability constraints, able to bypass current defences. Moreover, this analysis will enable us to understand the limitations of the defence mechanisms, and to propose novel defences, based on the uncertainty of the model, to help mitigate these attacks.

Chapter 5

Stealthy Attacks Against Regression and Defences

Synopsis

We propose a novel optimal attack formulation against regression models that considers different degrees of detectability constraints to bypass state-of-the-art defences, and a new methodology based on normalisation of objectives to solve the problem and obtain different trade-offs between effectiveness and detectability. We also develop a defence—called BayesClean—built upon a Bayesian linear regression to detect strong and stealthy attacks, which improves on previous defences the attack is stealthy and when the number of poisoning points is large.

5.1 Introduction

.

As mentioned in Ch. 4, regression learning is widely used in mission-critical systems: for instance, in the monitoring of aircraft engines (Chati and Balakrishnan, 2017), in medicine for the development of pharmaceuticals (Ekins et al., 2019), hedge funds management (Meligkotsidou et al., 2009), and in industrial processes for predictive maintenance (Wen et al., 2022) and quality control (Sun and Hou, 2017). Also in regression settings, the training data is often collected from sources that are partially trusted or easy to compromise such as IoT devices or from a distributed data supply chain. Despite this,

the robustness of regression models to poisoning attacks has not been widely addressed in the literature. Most of the studies that do address it (Mei and Zhu, 2015; Zhang et al., 2018; Jagielski et al., 2018; Müller et al., 2020; Wen et al., 2021; Weerasinghe et al., 2021; Li et al., 2021), assume that the attacker's objective is to maximise the damage caused by the attack, without properly considering detectability constraints that can help evade outlier detectors. The defences proposed in these studies mitigate such attacks to various extents, which we further analyse below.

However, as mentioned in Ch. 4, this threat model is fundamentally ill-posed. Whilst an attacker does aim to maximise the effectiveness of the attack, they also aim to remain undetected. If the attack can be easily detected, the training set may be discarded in part or in full, thus thwarting the objective of the attack entirely. If the attacker can remain undetected, the attack may be less effective, but nevertheless significantly damage the model.

In this chapter, we shall answer the following research questions (Ch. 1), in the context of regression:

- **RQ1**: Can we craft attacks with detectability constraints capable of modelling different trade-offs between effectiveness and detectability, and bypassing current defences?
- **RQ2**: Can we design effective defences capable of mitigating strong and stealthy attacks?

To conduct this evaluation, it is possible to formulate the attack as a parameterisable combination of the two attacker's objectives: to maximise the attack effectiveness whilst also minimising the potential for the attack to be detected. Such a formulation of the attack, which we model as a multiobjective bilevel optimisation problem, has not been defined in the literature before and presents unique challenges. We propose such a formulation here and also discuss alternatives.

In particular, understanding the robustness of linear regression, as a fundamental regression algorithm, to adversaries can enable future research on other classes of supervised learning methods. This way, we can gain further intuitions about the possible attacks. Hence, we focus on linear regression models and evaluate attacks and defences using the same or similar datasets to those used in the literature so far to facilitate comparison.

Then, we also we discuss and evaluate the scalability aspects of our approach in DNNs. Indeed, the formulation proposed here will enable further studies on the robustness of regression models to more realistic attacks.

We shall show that the defences proposed so far are vulnerable to poisoning attacks that take into account attack detectability and compare the defences' performance. We propose a novel formalisation and methodology for poisoning regression models that combines attack objective and detectability. This allows us to carry out this evaluation and offers new insights into the factors that limit the defences' performance. We also discuss dataset factors, in particular noisy datasets—whose noise is not purposely crafted by the attacker—that affect attack performance. Our evaluation is conducted on linear regression models and DNNs on datasets that have also been used in the related work on this topic.

We also propose a new defence against poisoning attacks in regression models, which we call BayesClean. In essence, BayesClean builds upon a Bayesian linear regression model with hyperparameters learnt according to the training data. Then, BayesClean rejects points based on their location within the predictive variance of the model. BayesClean is robust to poisoning attacks that consider attack detectability and compares favourably with other defences when the number of poisoning points is large.

In summary, the contributions of this chapter are as follows:

- We propose a novel stealthy attack formulation against regression algorithms via multiobjective bilevel optimisation: one of the objectives accounts for attack effectiveness, while the other accounts for the detectability. This formulation is more realistic than previous approaches as in many applications users will deploy a detector to reject points that are suspicious.

- We propose a novel detectability-risk function and a novel approach to solving the multiobjective bilevel optimisation problem via normalisation of the objectives. This allows us to consider different degrees of detectability for the attacker.

- We propose a novel defence based on uncertainty, which we call BayesClean, that improves on state-of-the-art defences when attacks consider detectability constraints and when the number of poisoning points is large.

- We experimentally show that state-of-the-art defences do not mitigate these stealthy attacks, even if they are not designed as adaptive attacks targeting those specific

defences—in other words, without detectability constraints. We evaluate different defences proposed so far on real-world datasets, in both linear regression models and DNNs, and identify the factors that influence their effectiveness such as the use of a trusted validation set. To our knowledge, we are the first to analyse optimal data poisoning attacks targeting DNNs in regression settings.

- We empirically show that datasets that are more difficult for the ML model to fit to (e.g., because they are noisy), are also more difficult to defend. This supports the theoretical findings shown in Ch. 4, Sect. 4.4.
- We empirically show the effectiveness of BayesClean to mitigate strong and stealthy attacks, which evidences the importance of the model uncertainty to counter poisoning attacks.

5.2 Related Work

Most standard poisoning attacks against regression aim to maximise the MSE of the model, (Mei and Zhu, 2015; Zhang et al., 2018; Jagielski et al., 2018; Müller et al., 2020; Wen et al., 2021; Weerasinghe et al., 2021). Mei and Zhu (2015) poison convex learning algorithms, including linear regression, based on the concept of machine teaching (Zhu, 2013; Zhu et al., 2018). Jagielski et al. (2018) propose PK and LK poisoning attacks against linear regression which aims to increase the MSE of the model at test time. The PK optimal attack is based on a bilevel optimisation formulation solved by leveraging its KKT conditions (Karush, 1939; Kuhn and Tucker, 1951), similar to Xiao et al. (2015). Nevertheless, the computation of the hypergradients of the attacker is expensive due the inversion the Hessian w.r.t. the model's parameters. Jagielski et al. (2018) also propose an LK attack, called StatP, that samples points from a multivariate Gaussian distribution estimated from the clean training data. Then, StatP rounds the feature variables to the bounds of their feasible domain, queries the model and rounds the target variable to the opposite corner. However, these attacks can be detected by defences that reject extreme values. Weerasinghe et al. (2021) incrementally extends the optimal formulation of Jagielski et al. (2018) to linear regression models that use non-linear basis functions. Wen et al. (2021) also slightly modify the optimal poisoning attack algorithm of Jagielski et al. (2018) through including the influence of previous poisoning points in the attacker's objective. Müller et al. (2020) propose an LK

label-manipulation attack with a surrogate training set, flipping the labels to one of two possible thresholds. On the other hand, the attacker's objective for poisoning a regression model may also differ. For example, an attacker may want to increase or decrease the predicted value, or may want to maximise the dispersion of the training set, as explained in Ch. 4. Attacks targeting specific parameters of a linear regression model have also been considered (Li et al., 2021).

In terms of defences, in robust statistics, estimation procedures are available for robust regression that are not strongly affected by the presence of stochastic noise and outliers (Huber, 1964; Fischler and Bolles, 1981). However, these methods fail in high dimensions and can struggle to identify carefully crafted poisoning samples (Xu et al., 2012). Liu et al. (2017) design robust linear regression algorithms under the assumption that the feature matrix has low rank and can be projected to a lower dimensional space. Diakonikolas et al. (2019) propose SEVER, a robust meta-algorithm that can detect poisoning samples through outlier scores based on the singular value decomposition (SVD) of the gradients. Weerasinghe et al. (2021) propose BIG-LID, a computationally expensive LK defence for non-linear regression learning based on the concept of local intrinsic dimensionality (LID)[1] (Houle, 2017) to identify and reduce the impact of poisoning samples. All these methods have provable robustness guarantees, but the assumptions on which they rely are not usually satisfied in practical settings where attackers consider detectability constraints.

Other relevant defences include TRIM (Jagielski et al., 2018) and Proda (Wen et al., 2021). TRIM (Jagielski et al., 2018) iteratively finds the subset of the training data that minimises the model's loss, and uses this subset as a representative set to train the model. Similar to aggregation methods, Proda (Wen et al., 2021) filters poisoning points by bootstrapping equal-sized groups of training points. Each bootstrap set is independently subjected to linear regression, resulting in β trained models. Then, for each trained model, Proda selects the subset of the whole training set with the lowest errors on the trained model, which is subjected again to linear regression. Amongst these β trained models, Proda selects the model whose MSE is lower. Finally, Debugging Using Trusted Items (DUTI) (Zhang et al., 2018) identifies samples that have the biggest influence on the validation error on a small trusted clean set.

[1]The LID value of a data point identifies the minimum number of latent variables required to represent that data point and recent evidence suggests that LID is an indicator of the degree of being an outlier (Houle et al., 2018; Weerasinghe et al., 2022).

As a final remark, it is noteworthy that most of these defences assume they know the fraction of poisoning samples in the training dataset, which is unrealistic in most practical scenarios. Besides, deviations on the estimation of the fraction of poisoning points can have a negative impact on the model's performance for both poisoned and clean models. In contrast, in *BayesClean*, the defence we propose in this chapter, we do not need to know the fraction of poisoning points to reject, since we reject points according to the predictive variance of a Bayesian linear regression model.

5.3 Attack with Soft Detectability Constraints

We now turn our discussion to our proposed attack formulation with detectability constraints. We assume that the attacker can arbitrarily manipulate all the features and the label of the poisoned points, provided that the resulting points are within a feasible domain of valid data points. Again, this is consistent with scenarios in which an attacker has compromised part of the infrastructure from which data is collected. Having compromised, e.g., a sensor, the attack is only limited in the value reported by that sensor by the domain of feasible values.

We first introduce an optimal attack formulation that generalises the one used in related work (Subsect. 5.3.1). Then we present a novel formulation that also includes the detectability objective (Subsect. 5.3.2) and discuss practical considerations for this formulation. We discuss choices for the detectability objective (Subsect. 5.3.2) and outline our approach for solving the resulting multiobjective optimisation problem (Subsect. 5.3.3).

5.3.1 Optimal Attack Formulation

We consider regression settings. Given the input space $\mathcal{X} \subseteq \mathbb{R}^m$ and the continuous label space, $\mathcal{Y} \subseteq \mathbb{R}$, the learner aims to estimate the mapping $f : \mathcal{X} \to \mathcal{Y}$. Given a training set $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}_{\text{tr}_i}, y_{\text{tr}_i})\}_{i=1}^{n_{\text{tr}}} = (\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ with n_{tr} IID samples drawn from the underlying probability distribution $p(\mathcal{X}, \mathcal{Y})$, we can estimate f with a model $\mathcal{M}$ trained by minimising an objective function $\mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w})$ w.r.t. its parameters, $\mathbf{w} \in \mathbb{R}^d$.

Let us define the feature matrix $\mathbf{X}$, where each row is a features sample $\mathbf{x}_i \in \mathbb{R}^m$, and the output vector $\mathbf{y}$, where each element is the corresponding output sample $y_i \in \mathbb{R}$. Let us also define $\mathbf{X}_e = [\mathbf{1} \quad \mathbf{X}]^\mathsf{T}$, $\mathbf{x}_{e_i} = [1 \quad \mathbf{x}_i^\mathsf{T}]^\mathsf{T}$ and $\mathbf{w} = \begin{bmatrix} w_0 & \mathbf{w}_{1:M-1}^\mathsf{T} \end{bmatrix}^\mathsf{T}$, where $\mathbf{1}$ is a vector

of ones and w_0 denotes the bias term. The additive model assumed for linear regression is:

$$y_i = \mathbf{x}_{e_i}^\top \mathbf{w} + \epsilon_i, \quad i = 1, \ldots, n, \tag{5.1}$$

where the additive noise ϵ_i follows a Gaussian distribution and n is the number of samples. In a poisoning attack, the adversary aims to inject n_p poisoning points, $\mathcal{D}_\mathrm{p} = \{(\mathbf{x}_{\mathrm{p}_k}, y_{\mathrm{p}_k})\}_{k=1}^{n_\mathrm{p}} = (\mathbf{X}_\mathrm{p}, \mathbf{y}_\mathrm{p})$, in the training set to maximise some arbitrary objective, $\mathcal{A}$, evaluated on a set of target data points $\mathcal{D}_\mathrm{target}$. Then, the L_2-regularised training loss function of the model when an attacker injects the poisoning set, $\mathcal{D}_\mathrm{p}$, is:

$$\mathcal{L}(\mathcal{D}_\mathrm{tr} \cup \mathcal{D}'_\mathrm{p}, \mathbf{w}) = \frac{1}{2(n_\mathrm{tr} + n_\mathrm{p})} \left(||\mathbf{y}_\mathrm{tr} - \mathbf{X}_{\mathrm{tr}_e} \mathbf{w}||_2^2 + ||\mathbf{y}_\mathrm{p} - \mathbf{X}_{\mathrm{p}_e} \mathbf{w}||_2^2 + \lambda ||\mathbf{w}_{1:(d-1)}||_2^2 \right), \tag{5.2}$$

where λ is the regularisation hyperparameter, which controls the strength of the regularisation term, and $\mathbf{w}_{1:(d-1)}$ denotes all the entries of $\mathbf{w}$ but the bias term.[2]

An *optimal* poisoning attack, i.e., that maximises the impact on the model, can then be formulated as a bilevel optimisation problem:

$$\begin{aligned} \max_{\mathcal{D}'_\mathrm{p} \in \Phi(\mathcal{D}_\mathrm{p})} \quad & \mathcal{A}(\mathcal{D}_\mathrm{target}, \mathbf{w}^\star) \\ \text{s.t.} \quad & \mathbf{w}^\star \in \arg\min_{\mathbf{w} \in \mathcal{W}} \mathcal{L}(\mathcal{D}_\mathrm{tr} \cup \mathcal{D}'_\mathrm{p}, \mathbf{w}), \end{aligned} \tag{5.3}$$

where the outer problem is the attacker's objective (which does not explicitly depend on the poisoning points, but implicitly through the poisoned parameters, $\mathbf{w}^\star$), and the inner problem is the defender's problem, i.e., the training problem on the poisoned set. When optimising the poisoning points, we assume that the defender trains on all the data points $\mathcal{D}'_\mathrm{p}$ that lie in the feasible set $\Phi(\mathcal{D}_\mathrm{p})$, i.e., they do not discard points.

This formulation encompasses both *indiscriminate attacks*, where $\mathcal{D}_\mathrm{target} = \mathcal{D}_\mathrm{val}$, and the validation set is representative of the underlying clean data distribution, and *targeted attacks*, where $\mathcal{D}_\mathrm{target}$ represents a specific subset of target data points.[3] Generally, $\mathcal{A}$ represents the function for the attacker objective. In our case, as we aim to maximise the error in the predictions, we can use the MSE, i.e., the loss typically minimised by most

[2] As in Ch. 3, note that we do not regularise the bias term of the model. In this regard, we do not induce too much variance by leaving the bias term not regularised. Also, regularising the bias term can introduce a significant amount of underfitting (Goodfellow et al., 2016). If the bias were regularised, intuitively, the model would try to bring the bias term closer to the origin in order to minimise the loss function.

[3] Note that targeted attacks against linear regression also have an impact on other points that are not in the attacker's target, as we showed in Ch. 4.

regression models. Then, as we focus on indiscriminate attacks we use $\mathcal{A}(\mathcal{D}_{\text{target}}, \mathbf{w}^\star) = \mathcal{L}(\mathcal{D}_{\text{val}}, \mathbf{w}^\star)$ to model the attacker's objective.

Given the attacker's budget, n_{p}, each poisoning point can be optimised by solving Eq. (5.3) incrementally (point by point), in mini-batches, or for all poisoning points at the same time. In the two former cases, $\mathcal{D}_{\text{tr}}$ contains the previous poisoning points, which influences the computation of subsequent points. This formulation is often encountered in related studies on poisoning attacks both for regression and classification (Biggio et al., 2012; Xiao et al., 2015; Mei and Zhu, 2015; Muñoz-González et al., 2017; Jagielski et al., 2018; Müller et al., 2020; Wen et al., 2021; Weerasinghe et al., 2021; Carnerero-Cano et al., 2021, 2023).

5.3.2 Optimal Attack Formulation with Soft Detectability Constraints

As well as causing maximum damage, an attacker also seeks to remain undetected, as detection may compromise all the effort spent towards the attack. So a second, but equally important objective for the attacker is to reduce the *detectability risk*. This may blunt the effectiveness of the attack but is necessary to not jeopardise it entirely. The attacker's perception of the risk of being detected places thus a constraint upon the effectiveness of the attack and restricts the manipulation of the features and labels of the poisoning points. In other words, the detectability risk objective typically aims to decrease the rate of poisoning points that are detected as malicious and the rate of clean points that are detected as benign.

There is no general agreement on how to define detectability risk (also referred to as detectability constraints). *Hard* constraints,[4] typically considered by the adversarial ML community (Suciu et al., 2018; Demontis et al., 2019; Huang et al., 2020, 2021; Geiping et al., 2021a; Fowl et al., 2021b; Koh et al., 2022; Cinà et al., 2023), must be satisfied by any feasible solution. However, in practice, the attacker's willingness to reduce the effectiveness of the attack to avoid detection depends on the attacker's risk appetite. So a *soft* (parameterisable) formulation of the *detectability risk*, $\mathcal{R}$ is desirable to incorporate a continuous constraint in the optimisation process of the poisoning points. These soft

[4]In an optimisation problem, a hard constraint is a constraint that must be satisfied by any feasible solution to the model. On the other hand, a soft constraint can be violated, but violating the constraint incurs a penalty in the objective function (often, the greater the amount by which the constraint is violated, the greater the penalty) (Meseguer et al., 2006).

constraints also allow to model different trade-offs between effectiveness and detectability that are easy to interpret, and therefore they have been adopted in this chapter.

We revisit the bilevel formulation for the optimal attack to include the detectability risk. We define the outer objective as a multiobjective optimisation problem where one of the objectives is the MSE of the model (effectiveness of the attack), and the other is the detectability risk $\mathcal{R}$. Therefore, the attacker's problem can be defined as a *multiobjective bilevel optimisation problem*:

$$\begin{aligned} \max_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} \mathcal{L}(\mathcal{D}_{\mathrm{val}}, \mathbf{w}^\star), \min_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} \mathcal{R} \\ \text{s.t.} \quad \mathbf{w}^\star \in \arg\min_{\mathbf{w} \in \mathcal{W}} \mathcal{L}(\mathcal{D}_{\mathrm{tr}} \cup \mathcal{D}'_{\mathrm{p}}, \mathbf{w}). \end{aligned} \tag{5.4}$$

Effectiveness and detectability are conflicting objectives. So, a solution to the multiobjective problem must quantify the trade-off between them. A straightforward way to do so is using scalarisation (Marler and Arora, 2004), i.e., converting a multiobjective problem into a set of single-objective problems, each one characterised by a (set of) hyperparameter(s). That is, varying the "importance" of each objective by a scalar $\alpha \in [0, 1]$, also known as weighted sum:

$$\begin{aligned} \max_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} \quad & \mathcal{A}_d(\mathcal{D}'_{\mathrm{p}}) = \alpha \mathcal{L}(\mathcal{D}_{\mathrm{val}}, \mathbf{w}^\star) - (1 - \alpha)\mathcal{R} \\ \text{s.t.} \quad & \mathbf{w}^\star \in \arg\min_{\mathbf{w} \in \mathcal{W}} \mathcal{L}(\mathcal{D}_{\mathrm{tr}} \cup \mathcal{D}'_{\mathrm{p}}, \mathbf{w}), \\ & \alpha \in [0, 1]. \end{aligned} \tag{5.5}$$

The hyperparameter α is assigned by the attacker based on its risk appetite. A parametric sweep for α can thus investigate different trade-offs between effectiveness and detectability. Note that when combining objectives in this way, the correlation (redundancy) between objective functions is known to affect the search space structure and algorithm performance. The objectives should be neither not correlated at all nor very highly correlated, otherwise it can be difficult to achieve a smooth transition for this trade-off. In the extreme case where $\mathcal{R} \propto -\mathcal{L}$, the multiobjective problem reduces to a single-objective problem where it is not possible to find a trade-off.

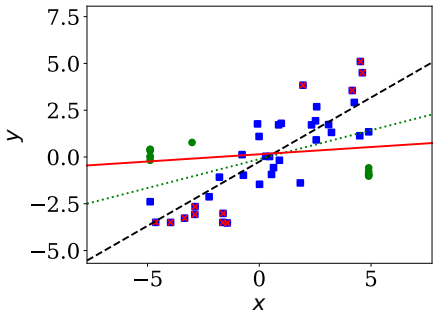


Figure 5.1: Synthetic example showing the effect of constraining poisoning points against TRIM (Jagielski et al., 2018). The training dataset contains 37 clean points and 13 poisoning points (26% of the training set). The features of the clean points are generated according to a uniform distribution $\mathcal{U}(-5, 5)$, the labels are generated according to the transformation: $y = 0.8x$, and the additive noise of the labels follows a Gaussian distribution $\mathcal{N}(\epsilon|0.0, 1.2^2)$. The linear regression model and TRIM are trained using SGD with a learning rate $\eta = 0.1$ and # epochs $= 40$ (full batch). The blue points represent the clean points, the green points are the poisoning points, the red crosses are the points rejected by TRIM, the black dashed line is the regression line learnt under the original clean data, the green dotted line is the regression line learnt under the complete poisoned training data, and the red solid line is the regression line learnt under the training data not rejected by TRIM. Because the poisoning points are closer to the clean points, TRIM fails to detect them.

Modelling the Detectability-Risk Function

To evaluate the detectability risk, i.e., the attacker's perception of the risk of being detected, different formulations are possible, as there are different ways in which a defender can try to detect poisoning point.

- **Data distribution-based detectability**: the poisoning points can be detected based on an estimation of the underlying data distribution, regardless of the target model. For instance, detection can based on distance functions between points: $\mathcal{R}(\mathcal{D}'_{\rm p}, \mathcal{D}_{\rm tr}, \mathcal{D}_{\rm val})$.

- **Model-based detectability**: the poisoning points can also be detected based on the effect that they have on the target model, for example, the impact that the attack has on the loss function. In this case, the detectability risk, $\mathcal{R}(\mathcal{D}'_{\rm p}, \mathbf{w}^\star, \mathbf{w}^\star_{\rm tr})$, is a function of the feasible poisoning points, $\mathcal{D}'_{\rm p}$, the poisoned parameters, $\mathbf{w}^\star$, and the clean parameters, $\mathbf{w}^\star_{\rm tr} \in \arg\min_{\mathbf{w}\in\mathcal{W}} \mathcal{L}(\mathcal{D}_{\rm tr}, \mathbf{w})$.

We choose the latter approach in this chapter, also noting that state-of-the-art defences reject points whose loss or norm of their gradients is large, e.g., (Jagielski et al., 2018; Diakonikolas et al., 2019; Wen et al., 2021). Such defences assume that the poisoning points are outliers and therefore easy to reject. They can fail when this is not the case.

For example, we can see in Fig. 5.1 that TRIM can fail by simply bounding the maximum values for the poisoning points.

Design of the Detectability-Risk Function

To calculate the detectability risk, an attacker can assume that the defender has access only to poisoned data and/or poisoned parameters to set their detection constraints. In this case, every new batch of poisoning points optimised shifts the detectability risk toward the distribution of the poisoning points, which allows to craft more aggressive points. This is optimistic and unrealistic in practice. A more conservative and more reasonable position for the attacker is to assume that the defender has access to clean parameters and/or clean data when setting its detection. In this case, the detectability risk does not change with the increasing number of poisoning points. It is this more realistic option that we have chosen.

In particular, one can design a detectability-risk function that precisely controls the loss of the poisoning points, for example by minimising the loss function on the poisoning points: $\mathcal{R}(\mathcal{D}'_{\mathrm{p}}, \mathbf{w}^{\star}_{\mathrm{tr}}) = \mathcal{L}(\mathcal{D}'_{\mathrm{p}}, \mathbf{w}^{\star}_{\mathrm{tr}})$. However, this function and the attacker's objective function, $-\mathcal{L}(\mathcal{D}_{\mathrm{val}}, \mathbf{w}^{\star})$, can be redundant, as they both favour points that are close to the poisoned regression hyperplane, whose impact is therefore limited. To reduce this redundancy, we redefine the detectability risk as the product of two loss functions that penalise points that are near the bounds of a certain "confidence interval":

$$\mathcal{R}(\mathcal{D}'_{\mathrm{p}}, \mathbf{w}^{\star}_{\mathrm{tr}}) = \mathcal{L}(\mathcal{D}'_{\mathrm{p}}, \mathbf{w}^{\star}_{\mathrm{tr}_+})\mathcal{L}(\mathcal{D}'_{\mathrm{p}}, \mathbf{w}^{\star}_{\mathrm{tr}_-}). \tag{5.6}$$

The parameters $\mathbf{w}^{\star}_{\mathrm{tr}_+}$ and $\mathbf{w}^{\star}_{\mathrm{tr}_-}$ are all equal to the clean parameters $\mathbf{w}^{\star}_{\mathrm{tr}}$ except their bias terms: $b^{\star}_{\mathrm{tr}_+} = b^{\star}_{\mathrm{tr}} + \sigma$ and $b^{\star}_{\mathrm{tr}_-} = b^{\star}_{\mathrm{tr}} - \sigma$; σ is the standard error of estimate (a.k.a. residual standard deviation/error):

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n_{\mathrm{tr}}} \left(y_{\mathrm{tr}_i} - \mathcal{M}(\mathbf{x}_{\mathrm{tr}_i})\right)^2}{df}}, \tag{5.7}$$

where df are the degrees of freedom of the residuals, which in the case of linear regression are: $n_{\mathrm{tr}} - p - 1$.

This detectability function (5.6) allows a smoother transition between the two objectives. We shall justify and analyse this function in more detail in Ch. 6, Subsect. 6.7. Hence, by

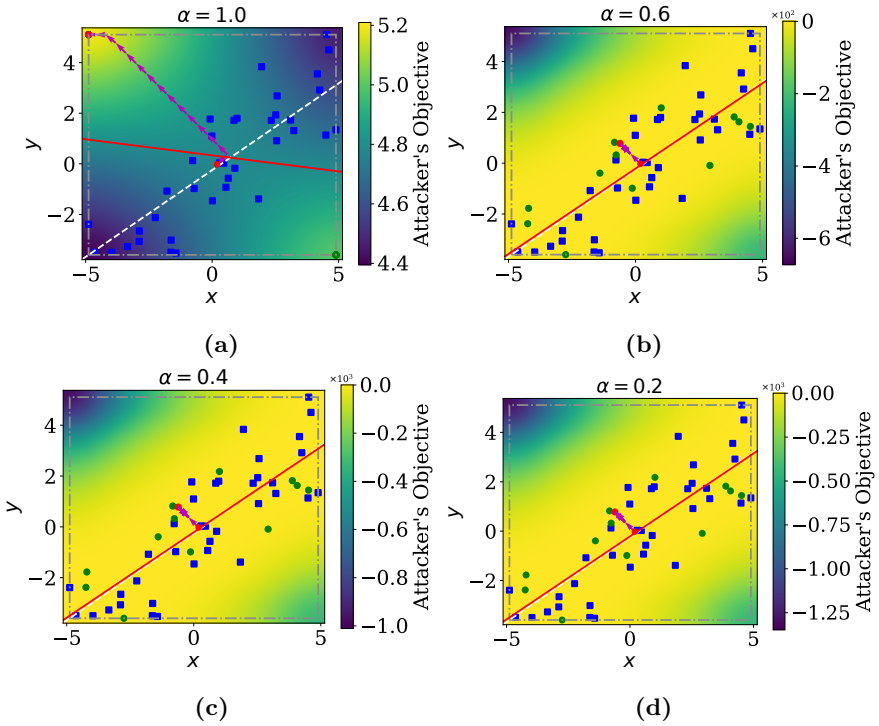


Figure 5.2: Effect of not normalising the objective functions in a synthetic example. The blue points represent the clean points, the green points are the poisoning points, and the two red points represent a poisoning point at the beginning and at the end of its optimisation process, respectively. Its trajectory during the optimisation process is depicted by the magenta line. The white dashed line is the regression line learnt under the original clean data, and the red solid line is the regression line learnt under the complete poisoned training data. The colourmap represents the value of the attacker's objective as a function of the location of the poisoning point. The experimental settings are the same as in Fig. 5.1. (a) $\alpha = 1$. (b) $\alpha = 0.6$. (c) $\alpha = 0.4$. (d) $\alpha = 0.2$. Note the sudden transition for the poisoning points from outliers ($\alpha = 1$) to inliers ($\alpha = \{0.6, 0.4, 0.2\}$). This is because the detectability objective dominates the effectiveness objective.

sweeping values of α between 0 to 1, we obtain poisoning points that shift smoothly from inliers to outliers, balancing the two goals proposed: attack success and stealth. However, in practice, the objectives can be of a different order of magnitude, which can make one of the objectives dominate the other during the optimisation process. We therefore need to define a method to balance the two objectives, which we introduce next.

Practical Aspects: Normalisation of Objective Functions

As the effectiveness and the detectability objective functions used in Eq. (5.5) can have values of different scale, normalisation of the objectives is required to obtain solutions consistent with α. We shall also explain this behaviour in Ch. 6, Subsect. 6.7. This also provides interpretability for α, i.e., the stealth of the attack.

In particular, gradient-based approaches can be very sensitive to different value ranges

in the two objective functions. In this case, the gradients can have significantly different norms for one of the objectives compared to the other one, leading to one objective completely dominating the optimisation process Milojković et al. (2019). This phenomenon can be observed in Fig. 5.2, where there is a sudden transition from outliers ($\alpha = 1$) to inliers ($\alpha = \{0.6, 0.4, 0.2\}$) because the detectability risk dominates the effectiveness. This is an undesirable effect that we aim to mitigate with our normalisation algorithm.

Inspired by normalisation methods in multiobjective optimisation (Marler and Arora, 2004), we develop a heuristic to normalise the two objective functions. Our goal is to compute the normalisation coefficient for each objective, which we define as their respective maximum value.

As for the detectability-risk function, using a clean reference for the model's parameters (Eq. (5.6)) makes it easier to compute its maximum, $\max_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} \mathcal{A}_d(\mathcal{D}'_{\mathrm{p}})|_{\mathcal{L}=0}$. Because the parameters are clean, the objective does not depend on the optimisation of the previous poisoning points:

$$\mathcal{R}_{\mathrm{norm}} \approx \frac{\mathcal{R}}{\mathcal{R}_{\mathrm{ref}}} = \frac{\mathcal{R}}{\max_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} \mathcal{A}_d(\mathcal{D}'_{\mathrm{p}})|_{\mathcal{L}=0}}. \tag{5.8}$$

Hence, the detectability-risk function presents a similar value for all the batches of poisoning points computed. In practice, as described in Alg. 5.1, we empirically determine its normalisation coefficient as the maximum value of the detectability risk across these batches, when the poisoning points are unconstrained ($\alpha = 1$). The reason is that their corresponding values can slightly differ due to, e.g., the initialisation of the poisoning points and/or the initialisation of the model's parameters. Thus, this normalisation coefficient is computed for $\alpha = 1$, and then used for different values of α.

In the case of the effectiveness objective, the challenge is that the normalisation coefficient depends on the size of the batch of poisoning points that are optimised at the same time, and on whether the objective is measured on the clean or the poisoned model. It is therefore difficult to obtain the maximum value of the effectiveness objective, $\max_{\mathcal{D}'_{\mathrm{p}} \in \Phi(\mathcal{D}_{\mathrm{p}})} \mathcal{A}_d(\mathcal{D}'_{\mathrm{p}})|_{\mathcal{R}=0}$, as this value depends on the values of the previous batches of poisoning points computed, which may, in turn, be constrained. We use instead the following heuristic: we carry out a sequential optimisation of poisoning points—i.e., B_{p} poisoning points are optimised at the same time, so that there are $n_{B_{\mathrm{p}}} = \lceil n_{\mathrm{p}}/B_{\mathrm{p}} \rceil$ batches of poisoning points. Once a batch of poisoning points is optimised, it is fixed in the training set and the next batch is optimised. Therefore, we calculate a new normalisation

Algorithm 5.1: Computation of B_p poisoning points ($\alpha = 1$) and of the normalisation coefficient for the detectability-risk function

Input: $\mathcal{M}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{D}_\mathrm{val}$, $\mathcal{D}_\mathrm{tr}$, n_p, B_p, $\{\mathcal{P}^{(i)}\}_{i=1}^{n_{B_\mathrm{p}}}$, T_out, T, γ, η
Output: $\mathcal{R}_\mathrm{ref}$

$n_{B_\mathrm{p}} \leftarrow \lceil n_\mathrm{p}/B_\mathrm{p} \rceil$
$\alpha \leftarrow 1$ `// Set maximum effectiveness and minimum detectability for the reference`
$\mathcal{A}_{\mathrm{ref}_{\alpha=1}} \leftarrow 1$
$\mathcal{R}_{\mathrm{ref}_{\alpha=1}} \leftarrow 1$
for $i = 0$ **to** $n_{B_\mathrm{p}} - 1$ **do**
  $\mathbf{w}^{(0)} \leftarrow$ `initW`$(\mathcal{M})$ `// Initialise w`
  for $t = 0$ **to** $T - 1$ **do**
    $\mathbf{g} \leftarrow \nabla_\mathbf{w} \mathcal{L}(\mathbf{w}^{(t)})$
    $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \mathbf{g}$ `// Stochastic Gradient Descent`
  end for
  $\mathcal{D}_\mathrm{p}^{(i)} \leftarrow$ `projHypGradAsc`$(\mathcal{M}, \mathcal{L}, \mathcal{L}_{\mathrm{ref}_{\alpha=1}}, \mathcal{R}, \mathcal{R}_{\mathrm{ref}_{\alpha=1}}, \mathcal{D}_\mathrm{val}, \mathcal{D}_\mathrm{tr}, B_\mathrm{p}, \mathcal{P}^{(i)}, T_\mathrm{out}, T, \alpha, \gamma, \eta)$
  $\mathcal{R}_\mathrm{ref}^{(i)} \leftarrow \mathcal{R}(\mathcal{D}_\mathrm{p}^{(i)})$
  $\mathcal{D}_\mathrm{tr} \leftarrow (\mathcal{D}_\mathrm{tr} \setminus \{(\mathbf{x}_{\mathrm{tr}_k}, y_{\mathrm{tr}_k})\}_{k \in \mathcal{P}^{(i)}}) \cup \mathcal{D}_\mathrm{p}^{(i)}$ `// Replace` B_p `samples of` $\mathcal{D}_\mathrm{tr}$ `by` $\mathcal{D}_\mathrm{p}^{(i)}$
end for
$\mathcal{R}_\mathrm{ref} \leftarrow \max_i \{\mathcal{R}_\mathrm{ref}^{(i)}\}_{i=1}^{n_{B_\mathrm{p}}}$

coefficient for each batch of poisoning points. Then, for the ith batch of poisoning points, we re-normalise the effectiveness objective as follows:

$$\mathcal{L}_\mathrm{norm}\left(\mathcal{D}_\mathrm{val}, \mathbf{w}^{\star(i)}\right) \approx \frac{\mathcal{L}\left(\mathcal{D}_\mathrm{val}, \mathbf{w}^{\star(i)}\right)}{\mathcal{L}_\mathrm{ref}} \approx \frac{\mathcal{L}\left(\mathcal{D}_\mathrm{val}, \mathbf{w}^{\star(i)}\right)}{\mathcal{L}\left(\mathcal{D}_\mathrm{val}, \mathbf{w}^{\star(i-1)}\right)}, \tag{5.9}$$

where $\mathbf{w}^{\star(i-1)}$ are the poisoned parameters trained on the previous optimal poisoned set. When computing the first batch of poisoning points, the previous poisoned set is the clean training set, and therefore $\mathbf{w}^{\star(i-1)} = \mathbf{w}_\mathrm{tr}^\star$. Even though this heuristic does not calculate the maximum of the objective, it provides an empirical reference for the range of values that the effectiveness objective can take during its optimisation. This computation is included in Alg. 5.2. Therefore, different values of α lead to different normalisation coefficients for the effectiveness objective. The functionality of Alg. 5.2 is also similar to Alg. 5.1 and Alg. 3.2.

In summary, to solve the multiobjective bilevel optimisation problem (5.5), we first need to compute the normalisation coefficient for the detectability-risk function for $\alpha = 1$ (Alg. 5.1), and then, for $\alpha \neq 1$, we compute the corresponding normalisation coefficient for

Algorithm 5.2: Computation of B_p poisoning points ($\alpha \neq 1$) and of the normalisation coefficient for the effectiveness function

Input: $\mathcal{M}$, $\mathcal{L}$, $\mathcal{R}$, $\mathcal{R}_\mathrm{ref}$ $\mathcal{D}_\mathrm{val}$, $\mathcal{D}_\mathrm{tr}$, n_p, B_p, $\{\mathcal{P}^{(i)}\}_{i=1}^{n_{B_\mathrm{p}}}$, T_out, T, α, γ, η
Output: $\mathcal{D}_\mathrm{tr}$

$n_{B_\mathrm{p}} \leftarrow \lceil n_\mathrm{p}/B_\mathrm{p} \rceil$
for $i = 0$ **to** $n_{B_\mathrm{p}} - 1$ **do**
  $\mathbf{w}^{(0)} \leftarrow \texttt{initW}(\mathcal{M})$ `// Initialise w`
  for $t = 0$ **to** $T - 1$ **do**
    $\mathbf{g} \leftarrow \nabla_\mathbf{w} \mathcal{L}(\mathbf{w}^{(t)})$
    $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \mathbf{g}$ `// Stochastic Gradient Descent`
  end for
  $\mathcal{L}_\mathrm{ref}^{(i)} \leftarrow \mathcal{L}(\mathcal{D}_\mathrm{val}, \mathbf{w}^{(T)})$
  $\mathcal{D}_\mathrm{p}^{(i)} \leftarrow \texttt{projHypGradAsc}(\mathcal{M}, \mathcal{L}, \mathcal{L}_\mathrm{ref}^{(i)}, \mathcal{R}, \mathcal{R}_\mathrm{ref}, \mathcal{D}_\mathrm{val}, \mathcal{D}_\mathrm{tr}, B_\mathrm{p}, \mathcal{P}^{(i)}, T_\mathrm{out}, T, \alpha, \gamma, \eta)$
  $\mathcal{D}_\mathrm{tr} \leftarrow (\mathcal{D}_\mathrm{tr} \setminus \{(\mathbf{x}_{\mathrm{tr}_k}, y_{\mathrm{tr}_k})\}_{k \in \mathcal{P}^{(i)}}) \cup \mathcal{D}_\mathrm{p}^{(i)}$ `// Replace` B_p `samples of` $\mathcal{D}_\mathrm{tr}$ `by` $\mathcal{D}_\mathrm{p}^{(i)}$
end for

the effectiveness function (Alg. 5.2). Fig. 5.3 shows the effect of normalising the objective functions in a synthetic example. We can see that the normalisation procedure enables a smoother transition as the value of α changes. Fig. 5.4 shows that crafting poisoning points close to the clean distribution ($\alpha = 0.4$) allows to shift the regression line even when TRIM is deployed as a defence. In contrast, optimal poisoning points without detectability constraints ($\alpha = 1$) can be easily filtered by TRIM.

5.3.3 Solving Optimal Poisoning Attacks with Soft Detectability Constraints

We have introduced a novel multiobjective bilevel optimisation formulation to evaluate attacks of different degrees of detectability. We have discussed the challenges of the multiobjective formulation and proposed a method to normalise the two objectives: effectiveness and detectability risk. The next step is then to solve the bilevel problem.

As we discussed in Ch. 3, Subsect. 3.3.2, solving the multiobjective bilevel optimisation problem in Eq. (5.5) is strongly NP-Hard (Bard, 2013) and, even if the inner problem is convex, the bilevel problem is, in general, non-convex. However, it is possible to use gradient-based approaches to obtain (possibly) suboptimal solutions, i.e., finding local optima for the problem in Eq. (5.5).

State-of-the art attacks against regression algorithms (Jagielski et al., 2018; Wen et al., 2021) are based on the implicit function theorem, which require training the whole learning

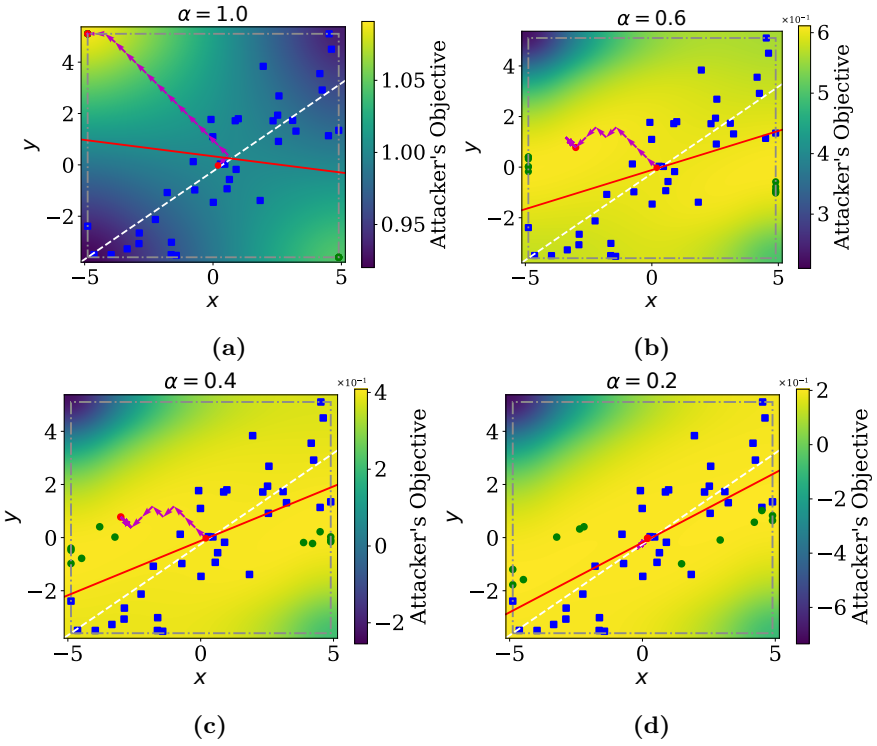
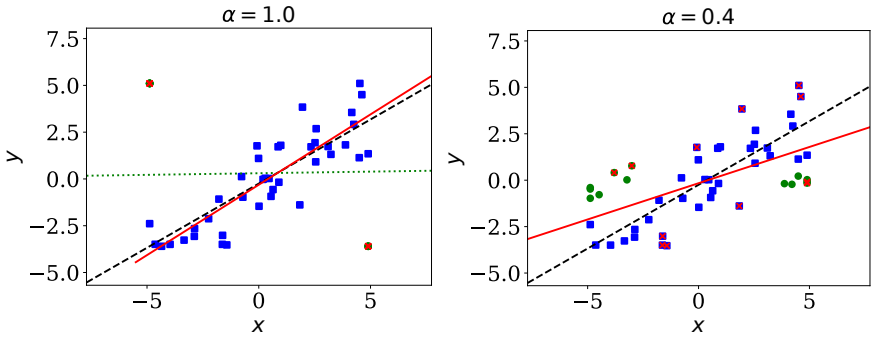


Figure 5.3: Effect of normalising the objective functions in a synthetic example. The blue points represent the clean points, the green points are the poisoning points, and the two red points represent a poisoning point at the beginning and at the end of its optimisation process, respectively. Its trajectory during the optimisation process is shown by the magenta line. The white dashed line is the regression line learnt under the original clean data, and the red solid line is the regression line learnt under the complete poisoned training data. The colourmap represents the value of the attacker's objective as a function of the location of the poisoning point. The experimental settings are the same as in Fig. 5.1. (a) $\alpha = 1$. (b) $\alpha = 0.6$. (c) $\alpha = 0.4$. (d) $\alpha = 0.2$. Compared to Fig. 5.2, in this case, normalising the objective functions allows a smoother transition for the poisoning points from outliers ($\alpha = 1$) to inliers ($\alpha = \{0.6, 0.4, 0.2\}$).

algorithm to compute the hypergradient, i.e., until the stationarity conditions are met. To sidestep this problem, different techniques have been proposed to estimate the value of the hypergradients (Domke, 2012; Maclaurin et al., 2015; Pedregosa, 2016; Franceschi et al., 2017; Muñoz-González et al., 2017; Franceschi et al., 2018; Grazzi et al., 2020). These techniques do not require retraining the learning algorithm each time the hypergradient is computed. Instead, they estimate the hypergradient by truncating the learning in the inner problem to a reduced number of training iterations.

As in Ch. 3, Subsect. 3.3.2, to compute the hypergradients for the outer objective, we assume that the first and second derivatives of the loss function, $\mathcal{L}$, and the first derivatives of the detectability risk, $\mathcal{R}$, are Lipschitz-continuous functions. As described in (Franceschi et al., 2017) and in Ch. 3, Subsect. 3.3.2, we can think of the training



(a) (b)

Figure 5.4: Synthetic example showing the effect of constraining poisoning points against TRIM (Jagielski et al., 2018), for $\alpha = 1$ and $\alpha = 0.4$. The blue points represent the clean points, the green points are the poisoning points, the red crosses are the points rejected by TRIM, the black dashed line is the regression line learned under the original clean data, the green dotted line is the regression line learned under the complete poisoned training data, and the red solid line is the regression line learned under the training data not rejected by TRIM. TRIM fails to detect the stealthy poisoning points, because they are closer to the clean distribution.

algorithm (inner problem) as a discrete-time dynamical system, described by a sequence of states $\mathbf{s}^{(t)} \in \mathbb{R}^{d_\mathrm{s}}$, with $t = 1, \ldots, T$, where each state depends on model's parameters, the accumulated gradients and/or the velocities, and the training data. We focus on SGD, i.e., $\mathbf{s}^{(t)} = \mathbf{w}^{(t)}$, so that each state of the sequence depends *only* on the previous state. We can therefore reformulate the bilevel problem in (5.5) as the constrained single-level optimisation problem:

$$\begin{aligned} \max_{\mathcal{D}'_\mathrm{p} \in \Phi(\mathcal{D}_\mathrm{p})} \quad & \mathcal{A}_d(\mathcal{D}'_\mathrm{p}) = \alpha \mathcal{L}(\mathcal{D}_\mathrm{val}, \mathbf{w}^{(T)}) - (1-\alpha)\mathcal{R} \\ \text{s.t.} \quad & \mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla_\mathbf{w} \mathcal{L}(\mathcal{D}_\mathrm{tr} \cup \mathcal{D}'_\mathrm{p}, \mathbf{w}^{(t-1)}), \\ & t = 1, \ldots, T, \\ & \alpha \in [0, 1], \end{aligned} \tag{5.10}$$

where η is the learning rate for SGD. Then, we estimate the hypergradients from the values of the parameters collected in the set of training states as[5]

$$\begin{aligned} \nabla_{\mathbf{X}_\mathrm{p}} \mathcal{A}_d(\mathcal{D}_\mathrm{p}) = & -(1-\alpha)\nabla_{\mathbf{X}_\mathrm{p}} \mathcal{R} \\ & + \left(\frac{d\mathbf{w}^{(T)}}{d\mathbf{X}_\mathrm{p}} \right)^\intercal \nabla_{\mathbf{w}_\mathrm{p}} \left(\alpha \mathcal{L}(\mathcal{D}_\mathrm{val}, \mathbf{w}^{(T)}) - (1-\alpha)\mathcal{R} \right), \end{aligned} \tag{5.11}$$

where the bottleneck is the computation of the implicit derivative $d\mathbf{w}^{(T)}/d\mathbf{X}_\mathrm{p}$. Given the

[5]The expression for $\mathbf{y}_\mathrm{p}$ is analogous.

Algorithm 5.3: RMD for multiobjective optimisation in regression poisoning

Input: $\mathcal{M}$, $\mathcal{A}$, $\mathcal{A}_{\text{ref}}$, $\mathcal{R}$, $\mathcal{R}_{\text{ref}}$, $\mathcal{L}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{tr}}$, $\mathcal{D}_{\text{p}}^{(\tau)}$, $\mathbf{w}^{(0)}$, T, η
Output: $\nabla_{\mathbf{X}_{\text{p}}}\mathcal{A}_d(\mathbf{w}^{(T)})$, $\nabla_{\mathbf{y}_{\text{p}}}\mathcal{A}_d(\mathbf{w}^{(T)})$

for $t = 0$ **to** $T-1$ **do**
  $\mathbf{g} \leftarrow \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}^{(t)})$
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta\mathbf{g}$ `// Stochastic Gradient Descent`
end for
$\mathcal{A}_{\text{norm}}(\mathbf{w}^{(T)}) \leftarrow \mathcal{A}(\mathbf{w}^{(T)})/\mathcal{A}_{\text{ref}}$
$\mathcal{R}_{\text{norm}}(\mathbf{w}^{(T)}) \leftarrow \mathcal{R}(\mathbf{w}^{(T)})/\mathcal{R}_{\text{ref}}$
$\mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)}) \leftarrow \alpha\mathcal{A}_{\text{norm}}(\mathbf{w}^{(T)}) - (1-\alpha)\mathcal{R}_{\text{norm}}(\mathbf{w}^{(T)})$
$d\mathbf{X}_{\text{p}}^{(T)} \leftarrow \nabla_{\mathbf{X}_{\text{p}}}\mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)})$
$d\mathbf{y}_{\text{p}}^{(T)} \leftarrow \nabla_{\mathbf{y}_{\text{p}}}\mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)})$
$d\mathbf{w}^{(T)} \leftarrow \nabla_{\mathbf{w}}\mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)})$
for $t = T-1$ **down to** 0 **do**
  $d\mathbf{g}_{\text{w}} \leftarrow \left(\nabla^2_{\mathbf{w}}\mathcal{L}(\mathbf{w}^{(t)})\right) d\mathbf{w}^{(t+1)}$ `// Hessian-vector product`
  $d\mathbf{g}_{\text{X}_{\text{p}}} \leftarrow \left(\nabla_{\mathbf{X}_{\text{p}}}\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}^{(t)})\right)^{\mathsf{T}} d\mathbf{w}^{(t+1)}$ `// Hessian-vector product`
  $d\mathbf{g}_{\text{y}_{\text{p}}} \leftarrow \left(\nabla_{\mathbf{y}_{\text{p}}}\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}^{(t)})\right)^{\mathsf{T}} d\mathbf{w}_{\text{p}}^{(t+1)}$ `// Hessian-vector product`
  $d\mathbf{w}^{(t)} \leftarrow d\mathbf{w}^{(t+1)} - \eta d\mathbf{g}_{\text{w}}$
  $d\mathbf{X}_{\text{p}}^{(t)} \leftarrow d\mathbf{X}_{\text{p}}^{(t+1)} - \eta d\mathbf{g}_{\text{X}_{\text{p}}}$
  $d\mathbf{y}_{\text{p}}^{(t)} \leftarrow d\mathbf{y}_{\text{p}}^{(t+1)} - \eta d\mathbf{g}_{\text{y}_{\text{p}}}$
end for
$\nabla_{\mathbf{X}_{\text{p}}}\mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)}) \leftarrow d\mathbf{X}_{\text{p}}^{(0)}$
$\nabla_{\mathbf{y}_{\text{p}}}\mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)}) \leftarrow d\mathbf{y}_{\text{p}}^{(0)}$

constraints in Eq. (5.10), it is obvious that the state $\mathbf{w}^{(t)}$ depends on the poisoning points both, directly by its expression, and indirectly through the previous state $\mathbf{w}^{(t-1)}$. Then, by applying the chain rule we obtain[5]

$$\frac{d\mathbf{w}^{(t)}}{d\mathbf{X}_{\text{p}}} = \frac{\partial\mathbf{w}^{(t)}}{\partial\mathbf{X}_{\text{p}}} + \frac{\partial\mathbf{w}^{(t)}}{\partial\mathbf{w}^{(t-1)}}\frac{d\mathbf{w}^{(t-1)}}{d\mathbf{X}_{\text{p}}}, \tag{5.12}$$

where $\partial\mathbf{w}^{(t)}/\partial\mathbf{w}^{(t-1)} = \mathbf{I}_d - \eta\nabla^2_{\mathbf{w}}\mathcal{L}$ and $\partial\mathbf{w}^{(t)}/\partial\mathbf{X}_{\text{p}} = -\eta\nabla_{\mathbf{X}_{\text{p}}}\nabla_{\mathbf{w}}\mathcal{L}$.

We include in Alg. 5.3 the RMD algorithm we use to compute the hypergradient estimate at the outer level problem (both for the features of the poisoning points, $\mathbf{X}_{\text{p}}$, and their labels, $\mathbf{y}_{\text{p}}$). For more details, including the algorithm complexity of RMD, see Subsect. 3.3.2, Alg. 3.1, and Table 3.1.

After computing the hypergradients, at each *hyperiteration* we use projected hyper-

Algorithm 5.4: Projected hypergradient ascent for regression poisoning

Input: $\mathcal{M}$, $\mathcal{A}$, $\mathcal{A}_{\text{ref}}$, $\mathcal{R}$, $\mathcal{R}_{\text{ref}}$, $\mathcal{L}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{tr}}$, B_{p}, $\mathcal{P}$, T_{out}, T, α, γ, η
Output: $\mathcal{D}_{\text{p}}^{(T_{\text{out}})}$

$\mathcal{D}_{\text{p}}^{(0)} \leftarrow \texttt{initDp}(\mathcal{D}_{\text{tr}}, B_{\text{p}})$ `// Generate` B_{p} `initial samples for` $\mathcal{D}_{\text{p}}^{(0)}$
$\mathcal{D}_{\text{tr}}^{(0)} \leftarrow (\mathcal{D}_{\text{tr}} \setminus \{(\mathbf{x}_{\text{tr}_k}, y_{\text{tr}_k})\}_{k \in \mathcal{P}}) \cup \mathcal{D}_{\text{p}}^{(0)}$ `// Replace` B_{p} `samples of` $\mathcal{D}_{\text{tr}}$ `by` $\mathcal{D}_{\text{p}}^{(0)}$
for $\tau = 0$ **to** $T_{\text{out}} - 1$ **do**
  $\mathbf{w}^{(0)} \leftarrow \texttt{initW}(\mathcal{M})$ `// Initialise` $\mathbf{w}$
  $\nabla_{\mathbf{X}_{\text{p}}} \mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)}), \nabla_{\mathbf{y}_{\text{p}}} \mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)}) \leftarrow$
  $\texttt{hypGrad}(\mathcal{M}, \mathcal{L}, \mathcal{L}_{\text{ref}}, \mathcal{R}, \mathcal{R}_{\text{ref}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{tr}}, \mathcal{D}_{\text{p}}^{(\tau)}, \mathbf{w}^{(0)}, T, \eta)$ `// RMD (Alg. 5.3)`
  $\mathbf{X}_{\text{p}}^{(\tau+1)} \leftarrow \Pi_{\Phi(\mathcal{D}_{\text{p}})}\left(\mathbf{X}_{\text{p}}^{(\tau)} + \gamma \nabla_{\mathbf{X}_{\text{p}}} \mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)})\right)$ `// Projected Hypergradient Ascent`
  $\mathbf{y}_{\text{p}}^{(\tau+1)} \leftarrow \Pi_{\Phi(\mathcal{D}_{\text{p}})}\left(\mathbf{y}_{\text{p}}^{(\tau)} + \gamma \nabla_{\mathbf{y}_{\text{p}}} \mathcal{A}_{d_{\text{norm}}}(\mathbf{w}^{(T)})\right)$ `// Projected Hypergradient Ascent`
  $\mathcal{D}_{\text{tr}}^{(\tau+1)} \leftarrow (\mathcal{D}_{\text{tr}}^{(\tau)} \setminus \mathcal{D}_{\text{p}}^{(\tau)}) \cup \mathcal{D}_{\text{p}}^{(\tau+1)}$ `// Update` $\mathcal{D}_{\text{tr}}^{(\tau)}$ `with` $\mathcal{D}_{\text{p}}^{(\tau+1)}$
end for

gradient ascent to update the poisoning points and the hyperparameters:

$$\begin{aligned} \mathbf{X}_{\text{p}} &\leftarrow \Pi_{\Phi(\mathcal{D}_{\text{p}})}(\mathbf{X}_{\text{p}} + \gamma \ \nabla_{\mathbf{X}_{\text{p}}} \mathcal{A}_d), \\ \mathbf{y}_{\text{p}} &\leftarrow \Pi_{\Phi(\mathcal{D}_{\text{p}})}(\mathbf{y}_{\text{p}} + \gamma \ \nabla_{\mathbf{y}_{\text{p}}} \mathcal{A}_d), \end{aligned} \tag{5.13}$$

where γ is the learning rate for the outer problem and $\Pi_{\Phi(\mathcal{D}_{\text{p}})}$ is the projection operator for the poisoning points, $\mathcal{D}_{\text{p}}$, so that their updated values are within their feasible domain, $\Phi(\mathcal{D}_{\text{p}})$. These projector operators are defined as $\Pi_{\Phi(\ \cdot\ )}(input) \triangleq \text{clip}(input, \inf \Phi(\ \cdot\ ), \sup \Phi(\ \cdot\ ))$, so that their updated values are within the corresponding feasible domains, $\Phi(\ \cdot\ )$. This procedure is summarised in Alg. 5.4, which is analogous to the one described in Subsect. 3.3.3 and Alg. 3.2.

5.4 BayesClean: A Defence Based on Uncertainty

Previously, we have seen that one of the reasons why some state-of-the-art defences fail is because the poisoning points are more constrained, so that there is no clear distinction between the clean and poisoning distribution. We hypothesise that another aspect that can make these defences less effective is that they rely on the assumption that the number of corrupted points is low. When the number of poisoning points increases, there is a certain breaking point where the reference for the true distribution is too much poisoned to be considered useful. For example, we cannot rely on just rejecting points based on their

errors on a single reference model. This assumes that the output of the model is correct, and therefore large errors of that model means anomalies. If the reference is poisoned, we cannot trust those large errors anymore because they could correspond to genuine samples. Moreover, some studies (Jagielski et al., 2018; Zhang et al., 2018; Diakonikolas et al., 2019; Wen et al., 2021) assume a ratio of poisoning points to reject, when proposing defences. However, this ratio can be difficult to estimate in practice.

To overcome these limitations, we develop a defence based on Bayesian linear regression, which we call *BayesClean*, which (i) does not assume a ratio of poisoning points to reject, and is more robust to (ii) larger ratios of poisoning points and (iii) poisoning points that are more constrained. At a high level, the defence rejects poisoning points outside of a certain "confidence interval" around the mean prediction of the Bayesian linear regression model. We shall describe first the fundamentals of this model.

5.4.1 Bayesian Linear Regression

Bayesian regression techniques can be used to include regularisation hyperparameters in the estimation procedure. The L_2 regularisation used in classification (see Ch. 3) and ridge regression is equivalent to finding a maximum a posteriori estimation under a Gaussian prior over the parameters $\mathbf{w}$ with precision λ_B. Thus, it is possible to treat it as a random variable to be estimated from the data. However, in contrast to the formulation developed in Ch. 3, here we do not use a trusted set.

As with standard linear regression, we assume that the target variable y is given by a function $\mathbf{x}^\mathsf{T}\mathbf{w}$ with additive Gaussian noise. Thus we can write the likelihood as:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I}_n), \tag{5.14}$$

where β is the precision (inverse variance) of the Gaussian noise. Bayesian linear regression estimates then a probabilistic model of the regression problem. The prior for the parameters is given by a Gaussian distribution governed by a precision hyperparameter, λ_B:

$$p(\mathbf{w}|\lambda_B) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda_B^{-1}\mathbf{I}_d). \tag{5.15}$$

As in (Tipping, 2001), the priors over λ_B and β are chosen to be gamma distributions, the

conjugate prior for the precision of the Gaussian:

$$p(\lambda_B|\lambda_1,\lambda_2) = \text{Gamma}(\lambda_B|\lambda_1,\lambda_2) = \frac{1}{\Gamma(\lambda_1)}\lambda_2^{\lambda_1}\lambda_B^{\lambda_1-1}\exp(-\lambda_2\lambda_B), \tag{5.16}$$

$$p(\beta|\beta_1,\beta_2) = \text{Gamma}(\beta|\beta_1,\beta_2) = \frac{1}{\Gamma(\beta_1)}\beta_2^{\beta_1}\beta^{\beta_1-1}\exp(-\beta_2\beta), \tag{5.17}$$

where $\Gamma(\lambda_1)$ is the gamma function, defined as: $\Gamma(\lambda_1) = \int_0^\infty u^{\lambda_1-1}e^{-u}\,du$. To make these priors non-informative (i.e., flat), these hyperparameters are usually chosen to be small values: e.g., $\lambda_1 = \lambda_2 = \beta_1 = \beta_2 = 10^{-6}$.

Having defined the prior, Bayesian inference proceeds by computing the joint posterior over all unknowns given the data, $p(\mathbf{w},\lambda_B,\beta|\mathbf{X},\mathbf{y})$, which can be decomposed using the parameters' posterior, $p(\mathbf{w}|\mathbf{X},\mathbf{y},\lambda_B,\beta)$, and the hyperparameters' posterior, $p(\lambda_B,\beta|\mathbf{X},\mathbf{y})$, as:

$$p(\mathbf{w},\lambda_B,\beta|\mathbf{X},\mathbf{y}) = p(\mathbf{w}|\mathbf{X},\mathbf{y},\lambda,\beta)p(\lambda_B,\beta|\mathbf{X},\mathbf{y}). \tag{5.18}$$

The hyperparameters' posterior will play a key role in our defence, since it allows to learn the optimal hyperparameters under the poisoning points. From Bayes' rule, the parameters' posterior can be computed as

$$p(\mathbf{w}|\mathbf{X},\mathbf{y},\lambda_B,\beta) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w},\beta)p(\mathbf{w}|\lambda_B)}{p(\mathbf{y}|\mathbf{X},\lambda_B,\beta)}. \tag{5.19}$$

Eq. (5.19) is a well-known result (Bishop, 2006) and is given by a Gaussian distribution:

$$p(\mathbf{w}|\mathbf{X},\mathbf{y},\lambda_B,\beta) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_{\text{post}},\boldsymbol{\Sigma}_{\text{post}}), \tag{5.20}$$

$$\boldsymbol{\Sigma}_{\text{post}} = \left(\beta\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda_B\mathbf{I}_d\right)^{-1}, \tag{5.21}$$

$$\boldsymbol{\mu}_{\text{post}} = \beta\boldsymbol{\Sigma}_{\text{post}}\mathbf{X}^\mathsf{T}\mathbf{y}. \tag{5.22}$$

Then, given a new test point, $\mathbf{x}_*$, predictions are made for its corresponding label y_*, in terms of the predictive posterior distribution:

$$\begin{aligned} p(y_*|\mathbf{x}_*,\mathbf{X},\mathbf{y}) &= \int\int\int p(y_*|\mathbf{x}_*,\mathbf{w},\beta)p(\mathbf{w},\lambda_B,\beta|\mathbf{X},\mathbf{y})d\mathbf{w}d\lambda_B d\beta \\ &= \int\int\int p(y_*|\mathbf{x}_*,\mathbf{w},\beta)p(\mathbf{w}|\mathbf{X},\mathbf{y},\lambda_B,\beta)p(\lambda_B,\beta|\mathbf{X},\mathbf{y})d\mathbf{w}d\lambda_B d\beta. \end{aligned} \tag{5.23}$$

These computations are not analytically tractable, and an effective approximation

is needed. To do so, we shall follow the *evidence approximation* framework (Gull, 1989; MacKay, 1992; Bishop, 2006). We can assume that the hyperparameter posterior distribution $p(\lambda_B, \beta|\mathbf{X}, \mathbf{y})$ is sharply peaked around its most-probable values $\hat{\lambda}_B$ and $\hat{\beta}$, on the basis that functions generated utilising the posterior mode values are near-identical to those obtained by sampling from the full posterior distribution (Tipping, 2001). Then, the predictive distribution is obtained simply by marginalising over $\mathbf{w}$ in which λ_B and β are fixed to the values $\hat{\lambda}_B$ and $\hat{\beta}$,

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \approx p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \hat{\lambda}_B, \hat{\beta}) = \int p(y_*|\mathbf{x}_*, \mathbf{w}, \hat{\beta}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \hat{\lambda}_B, \hat{\beta}) d\mathbf{w}. \tag{5.24}$$

Learning the hyperparameters thus becomes the search for the mode of the hyperparameters' log-posterior, i.e.:

$$\max_{\lambda_B, \beta} \log p(\lambda_B, \beta|\mathbf{X}, \mathbf{y}) \propto \log p(\mathbf{y}|\mathbf{X}, \lambda_B, \beta) + \log p(\lambda_B) + \log p(\beta). \tag{5.25}$$

The optimisation problem (5.25) can be solved by exploiting an expectation-maximisation (EM) formulation, as in (Tipping, 2001; MacKay, 1992).

At convergence of the hyperparameter learning procedure (5.25), we make predictions based on the posterior distribution over the parameters, conditioned on the learnt hyperparameters $\hat{\lambda}_B$ and $\hat{\beta}$, using Eq. (5.24). Making use of a well-known result (Bishop, 2006), the predictive distribution takes the form

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \hat{\lambda}_B, \hat{\beta}) = \mathcal{N}(y_*|\mu_*, \sigma_*^2), \tag{5.26}$$

with

$$\mu_* = \boldsymbol{\mu}_{\text{post}}^{\mathsf{T}} \mathbf{x}_*, \tag{5.27}$$

$$\sigma_*^2 = \hat{\beta}^{-1} + \mathbf{x}_*^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{post}} \mathbf{x}_*. \tag{5.28}$$

The first term in Eq. (5.28) represents the noise on the data, whereas the second term reflects the uncertainty associated with the parameters $\mathbf{w}$.

Algorithm 5.5: BayesClean

Input: $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, $\lambda_1 = \lambda_2 = \beta_1 = \beta_2 = 10^{-6}$, c_1, c_2, T_{EM}
Output: $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$

1: $\hat{\lambda}_B, \hat{\beta} \in \arg\max_{\lambda_B, \beta} \log p(\mathbf{y}|\mathbf{X}, \lambda_B, \beta) + \log p(\lambda_B) + \log p(\beta)$ `// Learn hyperparameters of the Bayesian linear regression model via EM, up to` T_{EM} `iterations`
2: $\boldsymbol{\Sigma}_{\text{post}} \leftarrow (\hat{\beta}\mathbf{X}^\top\mathbf{X} + \hat{\lambda}_B\mathbf{I}_d)^{-1}$, $\boldsymbol{\mu}_{\text{post}} \leftarrow \hat{\beta}\boldsymbol{\Sigma}_{\text{post}}\mathbf{X}^\top\mathbf{y}$ `// Update posterior distribution`
3: $\boldsymbol{\mu}_* \leftarrow \mathbf{X}\boldsymbol{\mu}_{\text{post}}$, $\boldsymbol{\sigma}_*^2 \leftarrow \hat{\beta}^{-1} + \text{diag}(\mathbf{X}\boldsymbol{\Sigma}_{\text{post}}\mathbf{X}^\top)$ `// Compute posterior predictive distribution for each input point`
4: $\mathcal{I}_1 \leftarrow \{i \mid |y_i| \leq |\mu_{*_i} + c_1\sigma_{*_i}|\}$ `// Indices of points to accept`
5: $\mathcal{I}_2 \leftarrow \{i \mid |\mu_{*_i} + c_1\sigma_{*_i}| < |y_i| \leq |\mu_{*_i} + c_2\sigma_{*_i}|\}$ `// Indices of points to flag`
6: $\mathcal{I}_3 \leftarrow \{i \mid |y_i| > |\mu_{*_i} + c_2\sigma_{*_i}|\}$ `// Indices of points to reject`

5.4.2 Predictive Variance under Data Poisoning

In the particular case where $\hat{\beta}$ is not learnt from the data but fixed, as additional data points are observed, the posterior distribution becomes narrower. As a consequence, it can be shown (Qazaz et al., 1997) that the predictive variance becomes smaller, i.e., $\sigma^2_{*n+1} \leq \sigma^2_{*n}$. In the limit where the number of training points is very large, $n \to \infty$, the second term in Eq. (5.28) goes to zero, and the variance of the predictive distribution arises solely from the additive noise governed by the hyperparameter $\hat{\beta}$. However, in data poisoning settings, the poisoning points do not follow the genuine data distribution. Thus, if β is learnt from the data, this result does not hold. To see this, let us have a look at the expression for the ith update of β^{-1} in the EM procedure (Tipping, 2001):

$$\beta^{-1^{(i)}} = \frac{\left|\left|\mathbf{y} - \mathbf{X}\boldsymbol{\mu}^{(i)}_{\text{post}}\right|\right|_2^2 + 2\beta_2}{n - \sum_{j=1}^{d}\gamma_j^{(i)} + 2\beta_1}, \tag{5.29}$$

where $\gamma_j^{(i)} \in [0, 1]$—also updated in the EM algorithm—can be interpreted as a measure of how "well-determined" its corresponding parameter w_j is by the data (MacKay, 1992). Thus, if the poisoning points follow a distribution different from the clean distribution, the error term $\left|\left|\mathbf{y} - \mathbf{X}\boldsymbol{\mu}^{(i)}_{\text{post}}\right|\right|_2^2$ can increase, which implies a larger β^{-1} and thus a larger predictive variance. This observation leads us to the presentation of our defence mechanism.

5.4.3 BayesClean

BayesClean builds upon a Bayesian linear regression model, as described in Alg. 5.5.

First, the method learns the optimal hyperparameters $\hat{\lambda}_B$ and $\hat{\beta}$, where T_{EM} is the maximum number of iterations of the EM algorithm. Then, we update the posterior distribution, and the posterior predictive distribution is computed for each input point, $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. In Alg. 5.5, $\text{diag}(\mathbf{X}\boldsymbol{\Sigma}_{\text{post}}\mathbf{X}^\mathsf{T})$ denotes the vector of entries of the main diagonal of $\mathbf{X}\boldsymbol{\Sigma}_{\text{post}}\mathbf{X}^\mathsf{T}$. Subsequently, the algorithm splits $\mathcal{D}$ into three mutually exclusive sets of points, according to their location around the predictive mean. The points closest to the predictive mean, whose indices we define as $\mathcal{I}_1 = \{i \mid |y_i| \leq |\mu_{*_i} + c_1\sigma_{*_i}|\}$, are detected as benign. Then, the points corresponding to the set $\mathcal{I}_2 = \{i \mid |\mu_{*_i} + c_1\sigma_{*_i}| < |y_i| \leq |\mu_{*_i} + c_2\sigma_{*_i}|\}$, are flagged for inspection, because they can damage the performance of the model. In our experimental evaluation, we reject those points. Finally, the points that lie far from the predictive mean, i.e., $\mathcal{I}_3 = \{i \mid |y_i| > |\mu_{*_i} + c_2\sigma_{*_i}|\}$, are rejected by our algorithm. Fig. 5.5 shows the intuition of BayesClean, and how the predictive variance increases with the number of poisoning points injected in the training set, as explained in Subsect. 5.4.2. In Fig. 5.5, the purple region corresponds to the condition $\mathcal{I}_1$, the red region corresponds to $\mathcal{I}_2$, and the white region corresponds to $\mathcal{I}_3$. Intuitively, the mismatch between the clean and the poisoning distribution can increase the predictive variance and thus the uncertainty in the predictions.

Two advantages of our defence is that we do not have to define the number of points to reject beforehand—which is difficult to estimate in practical scenarios—and we do not require a trusted set, in contrast to the method developed in Ch. 3. The constants $c_2 \geq c_1$ are defined beforehand, but we shall keep those constants fixed for any ratio of poisoning and across different datasets.

Regarding the complexity of the algorithm, the updates for the hyperparameters depend on computing the covariance matrix of the parameters' posterior, which requires an inverse operation. In practice, we leverage the SVD decomposition of the design matrix, $\mathbf{X}$. Efficient numerical methods scale in time as $\mathcal{O}(\min(nd^2, n^2d))$ and in memory storage as $\mathcal{O}(d^2)$, where d is the number of features (or basis functions if used) (Cline and Dhillon, 2006; Golub and Van Loan, 2013).

In Fig. 5.6 we compare TRIM and BayesClean in a synthetic example, when the number of poisoning points is large. In the case of TRIM, we assume the defender knows the number of poisoning points. We observe that BayesClean can filter poisoning points also for larger ratios of poisoning. In the case of TRIM, in contrast, even if the defender

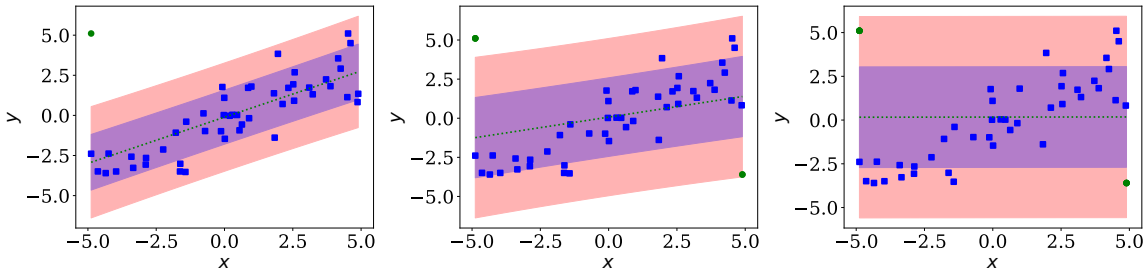


Figure 5.5: Behaviour of BayesClean in a synthetic example. In this example, $c_1 = 1$, $c_2 = 2$, $\lambda_1 = \lambda_2 = \beta_1 = \beta_2 = 10^{-6}$, and $T_{\text{EM}} = 300$ (maximum number of iterations when learning the hyperparameters). The blue points represent the clean points, the green points are the poisoning points, and the green dotted line is the regression line learnt under the complete poisoned training data. The purple region corresponds to the condition $\mathcal{I}_1 = \{i \mid |y_i| \leq |\mu_{*_i} + c_1\sigma_{*_i}|\}$, the red region corresponds to $\mathcal{I}_2 = \{i \mid |\mu_{*_i} + c_1\sigma_{*_i}| < |y_i| \leq |\mu_{*_i} + c_2\sigma_{*_i}|\}$. (a) 2% of poisoning points, and the white region corresponds to $\mathcal{I}_3 = \{i \mid |y_i| > |\mu_{*_i} + c_2\sigma_{*_i}|\}$. (b) 10% of poisoning points. (c) 18% of poisoning points. The predictive variance clearly increases with the number of poisoning points.

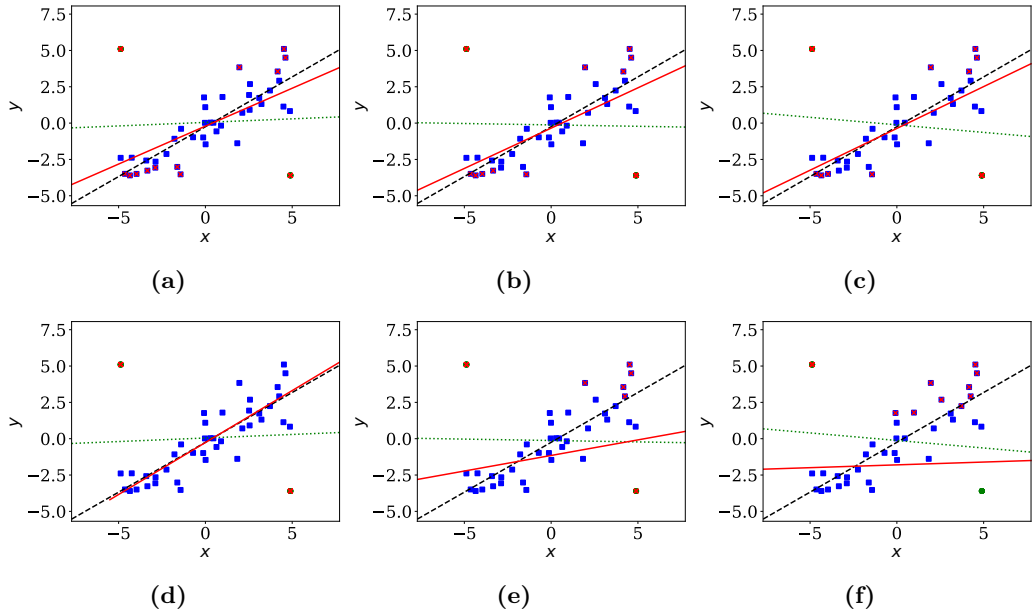


Figure 5.6: Comparison of BayesClean and TRIM in a synthetic example. The experimental settings are the same as in Fig. 5.1 and Fig. 5.5. The blue points represent the clean points, the green points are the poisoning points, the red crosses are the points rejected by the corresponding defence, the black dashed line is the regression line learnt under the original clean data, the green dotted line is the regression line learnt under the complete poisoned training data, and the red solid line is the regression line learnt under the training data not rejected by the corresponding defence. (a) BayesClean, 20% of poisoning points. (b) BayesClean, 24% of poisoning points. (c) BayesClean, 30% of poisoning points. (d) TRIM, 20% of poisoning points. (e) TRIM, 24% of poisoning points. (f) TRIM, 30% of poisoning points. We can observe that BayesClean is more robust than TRIM for large ratios of poisoning points.

knows the number of poisoning points, there is a certain number of poisoning points (around 24%) from which TRIM fails to detect some poisoning points. We recall that the

underlying assumption of TRIM is that poisoning points have a larger error than the clean points on the poisoned model. We hypothesise that TRIM fails because, for large ratios of poisoning points, the mean error of the clean points approaches or even surpasses the mean error of the poisoning points on the poisoned model, as the model's parameters are shifted towards the poisoning points. We will explore this hypothesis more formally in Ch. 7. In contrast, the properties of the predictive variance of BayesClean makes it more robust than TRIM for large ratios of poisoning points. In the experimental evaluation we will also show that BayesClean performs better than state-of-the-art defences when attacks consider detectability constraints.

5.5 Experimental Evaluation

In this section we empirically show that the novel attack formulation with soft detectability constraints proposed in Sect. 5.3 is capable of bypassing state-of-the-art defences. Moreover, we also show the effectiveness of the defence proposed in Sect. 5.4 to mitigate such attacks. As mentioned earlier in Subsect. 5.3.1, we evaluate indiscriminate attacks. We focus in our experimental evaluation on linear regression problems and DNNs. As for linear regression, this is in line with the related work available on poisoning regression models (Mei and Zhu, 2015; Zhang et al., 2018; Jagielski et al., 2018; Müller et al., 2020; Wen et al., 2021; Weerasinghe et al., 2021; Li et al., 2021). Thus, by focussing on linear regression we evaluate existing proposed defences in the context for which they have been proposed. Moreover, in contrast to most previous work on optimal poisoning attacks against regression models, which focusses on linear models, we also empirically evaluate the attack strategy in Eq. (5.5) against feed-forward DNNs. We provide an analysis of the robustness of the learning algorithms to worst-case scenarios for attacks with different strengths and detectability constraints. As explained in Ch. 3, note that the analysis of optimal indiscriminate poisoning attacks against non-convex models is substantially more computationally difficult.

In our results, we depict the test normalised mean squared error (NMSE) of the model when there is no defence deployed, computed as:

$$\text{NMSE}_{\text{test, no defence}} = \frac{\text{MSE}_{\text{test, no defence}}}{||\mathbf{y}_{\text{test}}||_2^2 / n_{\text{test}}}, \tag{5.30}$$

Dataset	# Training Samples	# Validation Samples
Loan	3,361	1,008
Heart Disease	1,599	480
Boston Housing	253	76
Appliances	9,868	2,960

Dataset	# Test Samples	# Features
Loan	2,353	88
Heart Disease	1,119	28
Boston Housing	177	13
Appliances	6,907	16

Table 5.1: Characteristics of the datasets used in the experiments.

where n_{test} is the number of test samples and $\mathbf{y}_{\text{test}}$ is the vector of test labels; and the *test defence gain* of a particular defence i, defined as:

$$\text{Test Defence Gain}_i[\%] = \frac{\text{NMSE}_{\text{test, no defence}} - \text{NMSE}_{\text{test,i}}}{\text{NMSE}_{\text{test, no defence}}} \cdot 100, \tag{5.31}$$

as a function of the number of poisoning points. This gain quantifies the decrease of the test NMSE when a particular defence is deployed, and provides a practical measure of the feasible poisoning error described in Ch. 4, Sect. 4.4. A negative gain (red region in the following plots) means that the defence is worsening the performance of the model.

5.5.1 Experimental Settings

We evaluate the effectiveness and stealth of poisoning attacks in stationary settings. Thus, we perform a random split of the datasets to simulate absence of concept drift (i.e., changes in the data distribution through time). If we were to perform a time-aware split, it would be very difficult to determine whether the effectiveness of our poisoning attack was due to an intrinsic weakness of the regression models or due to the natural evolution of data (Pendlebury et al., 2019). We use four real-world datasets: interest rate of loans (Loan)—made on the Lending Club peer-to-peer lending platform (Kan, 2013) (preprocessed as in (Jagielski et al., 2018)); heart disease (Heart Disease) (Weerasinghe et al., 2021); Boston housing prediction (Boston Housing) (Harrison Jr. and Rubinfeld, 1978); and appliances energy prediction (Appliances) (Candanedo, 2017) (preprocessed as in (Weerasinghe et al., 2021)). These are datasets used in the related work on poisoning attacks or in standard regression benchmarks. All datasets are drawn at random from the original joint pool of points. The details for each dataset are included in Table 5.1.

Dataset (Model)	T_{out}	γ	T	η	Poisoning Batch
Loan (Linear Regression)	120	0.9	30	0.1	252
Heart Disease (Linear Regression)	50	0.9	40	0.1	120
Boston Housing (Linear Regression)	100	0.9	40	0.1	19
Appliances (Linear Regression)	50	0.9	70	0.2	740
Loan (DNN)	150	0.9	90	0.1	252
Heart Disease (DNN)	80	0.9	100	0.1	120

Table 5.2: Experimental settings for the poisoning attack.

All our results are the average of 10 repetitions with different random data splits for training, validation and test sets. For each repetition, we normalise the features w.r.t. the mean and standard deviation of the clean training data, and we subtract the mean of the clean training labels from all the labels.[6] For all the attacks, we measure the average test NMSE for different attack strengths, where the number of poisoning points ranges from 0% to 45%. The size of the batch of poisoning points that are simultaneously optimised is included in Table 5.2. In this way, we simulate seven different ratios of poisoning ranging from 0% to 45%.

We simulate different ratios of poisoning points in a cumulative manner: Once the optimisation of the current batch of poisoning points and hyperparameters is finished,[7] this batch of poisoning points is fixed and the next batch of poisoning points is replaced into the remaining clean training set, whereas the hyperparameters are re-initialised, to carry out their corresponding optimisation. To accelerate their optimisation, the hypergradients for the poisoning points are normalised with respect to their ℓ_2-norm. We analyse the underlying reason for this in Ch. 6, Sect. 6.3.

The parameters of the linear regression models are always initialised with zeros, for all the datasets. The DNN models have two hidden layers with Leaky ReLU activation functions (negative slope $= 10^{-2}$) as follows: (# Features) $\times$ 32 $\times$ 8 $\times$ 1. In the DNN models, these parameters are initially filled with values according to Xavier Initialisation method (Glorot and Bengio, 2010), using a uniform distribution for all the parameters except the bias terms, which are initialised with a value of 10^{-2}.

As in Ch. 3, for all the experiments we make use of SGD to both update the parameters in the forward pass of RMD, and train the model when testing the attack (full batch

[6]The analysis of attacks when the normalisation process is also poisoned is left as future work.

[7]The criterion to finish the loop that optimises the variables of the outer level problem is given by the number of hyperiterations.

Dataset (Model)	η_{tr}	# Epochs
Loan (Linear Regression)	0.1	30
Heart Disease (Linear Regression)	0.1	40
Boston Housing (Linear Regression)	0.1	40
Appliances (Linear Regression)	0.2	70
Loan (DNN)	0.1	90
Heart Disease (DNN)	0.1	100

Table 5.3: Experimental settings for testing the attacks.

training). The choice of the number of iterations for the inner problem, T, depends on the model and the training dataset. Low values of T could lead to low-quality approximations for the hypergradient. As T increases, the solution of RMD approaches the exact (true) hypergradient, but at the risk of overfitting the outer objective in the bilevel optimisation problem (Franceschi et al., 2018). The details of the attack settings are shown in Table 5.2, whereas the ones for testing the attacks are in Table 5.3.

With respect to L_2 regularisation, we tested a methodology akin to (Jagielski et al., 2018) and (Friedman et al., 2010), where the authors use K-fold cross-validation to select the value of λ, and the clean data is used both for training and validation in an unbiased way. However, in our experiments in the datasets mentioned before, the value of the regularisation hyperparameter learnt following this approach was very low, so the performance was very similar to the one obtain without using regularisation. Therefore, in the following experiments we set $\lambda = 0$. For an extensive empirical analysis of the effect of regularisation in the poisoning attack, see Ch. 3. We will also theoretically analyse the asymptotic behaviour of the attacker's objective function when the model is over-regularised in Ch. 6.

We evaluate our stealthy attack against several benchmark defences: TRIM (Jagielski et al., 2018), Huber (Huber, 1964), and SEVER (Diakonikolas et al., 2019). Then, we compare our novel defence with other state-of-the-art defences: TRIM (Jagielski et al., 2018), SEVER (Diakonikolas et al., 2019), Proda (Wen et al., 2021), and DUTI (Zhang et al., 2018). One of the drawbacks of these defences is that they require defining the number of points to reject beforehand. Since we are testing attacks corresponding to up to 45% of poisoning points, we consider strong defences that reject 40% of the input points. We also recall that DUTI is guided through a small trusted clean set, which represents an advantage w.r.t. other defences.

We set the hyperparameters of the defences as follows: for BayesClean, we set $\lambda_1 = \lambda_2 = \beta_1 = \beta_2 = 10^{-6}$, $c_1 = 0.5$, $c_2 = 1.5$, and $T_{\text{EM}} = 300$ (maximum number of iterations when learning the hyperparameters). For TRIM, SEVER, and Proda, we use the same training hyperparameters as in Table 5.3. For Proda, we set $\epsilon_{\text{Proda}} = 10^{-5}$ (upper bound on the probability that no group of points belongs to the clean training set) and $\gamma_{\text{Proda}} = 5$ (size of each group of points), considering its time complexity. For DUTI, we set $\lambda_{\text{DUTI}} = 0.1$ (regularisation hyperparameter), $c_{\text{DUTI}} = 100$ (confidence level on all trusted items) and the number of trusted items—sampled from the validation set—as for 101 for Loan, 48 for Heart Disease, 8 for Boston Housing, and 296 for Appliances (ratio w.r.t. the training set size similar to (Zhang et al., 2018)). For Huber, we set $\lambda_{\text{Huber}} = 10^{-4}$ (regularisation hyperparameter), $T_{\text{Huber}} = 10^4$ (maximum number of iterations), and we explore values for ϵ_{Huber} (hyperparameter that controls the number of samples that should be classified as outliers) in the range $[1.1, 10]$.

All the experiments have been run on 2×11 GB NVIDIA GeForce® GTX 1080 Ti GPUs. The RAM memory is 64 GB (4×16 GB) Corsair VENGEANCE DDR4 3000 MHz. The processor (CPU) is Intel® Core™ i7 Quad Core Processor i7-7700k (4.2 GHz) 8 MB Cache.

5.5.2 Attacks with Detectability Constraints

We evaluate the defences for attacks with different levels of detectability, i.e., for different values of α balancing the trade-off between the effectiveness and the detectability objective: $\alpha = 1$ corresponds to attacks entirely dominated by effectiveness, $\alpha = 0.3$ correspond to attacks that aims to remain undetected but still aims to achieve a level of effectiveness, whilst $\alpha = 0.1$ corresponds to *stealthy* attacks predominantly concerned with remaining undetected.

In each plot the test defence gain with respect to the base-case no-defence scenario is reported as a function of the fraction of poisoning points. A negative test defence gain (the area shadowed in red in the graphs) shows that the defence is less effective than the unprotected model, i.e., the defence is not worth deploying. A positive gain shows that the defence manages to mitigate the attack to some extent.

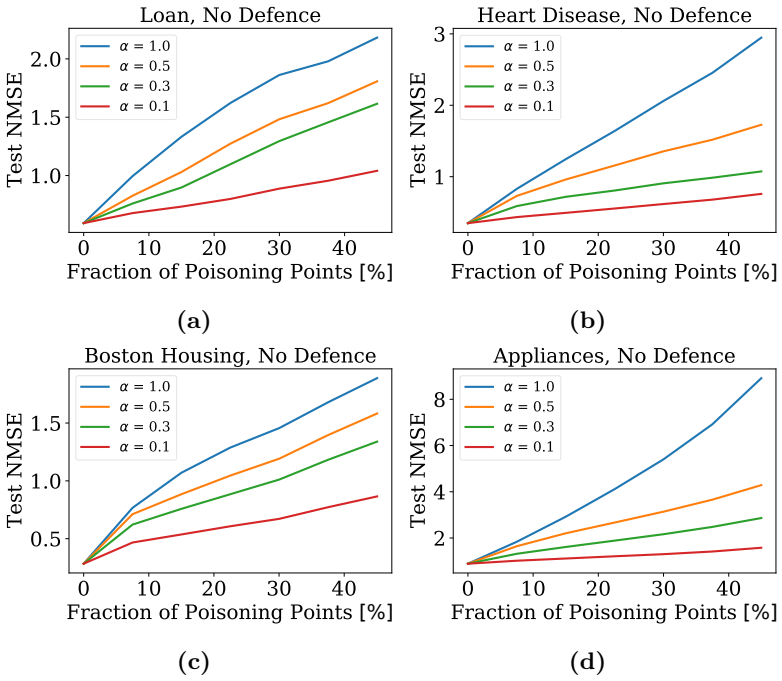


Figure 5.7: Test NMSE of linear regression when there is no defence deployed, for $\alpha = 1$, $\alpha = 0.5$, $\alpha = 0.3$, and $\alpha = 0.1$. (a) Loan. (b) Heart Disease. (c) Boston Housing. (d) Appliances.

Linear Regression

We fist report in Fig. 5.7 the results corresponding to the *base case* where defences have not been deployed. We can observe that the test NMSE increases with the value of α, as we generate more points that are out of distribution. In contrast, reducing α constrains more the poisoning points, with the goal of bypassing defences, as we shall see next.

The results showing the performance of the attack against several benchmark defences are presented in Fig. 5.8. Overall, these results show that current defences are generally not able to protect against poisoning attacks that consider detectability constraints. In particular, Huber performs very poorly for different values of α for all the datasets but Appliances. For $\alpha = 1$ we are considering attacks entirely dominated by the effectiveness objective, yet some defences fail and actually increase the test NMSE w.r.t. the base-case no-defence scenario. They typically have a beneficial effect only for low fractions of poisoning points and have a deleterious effect in all other cases. We shall analyse this phenomenon in Ch. 7. When the attack is very stealthy ($\alpha = 0.1$), for most defences the test defence gain is negative, therefore using the defence is more damaging than not using any defence at all.

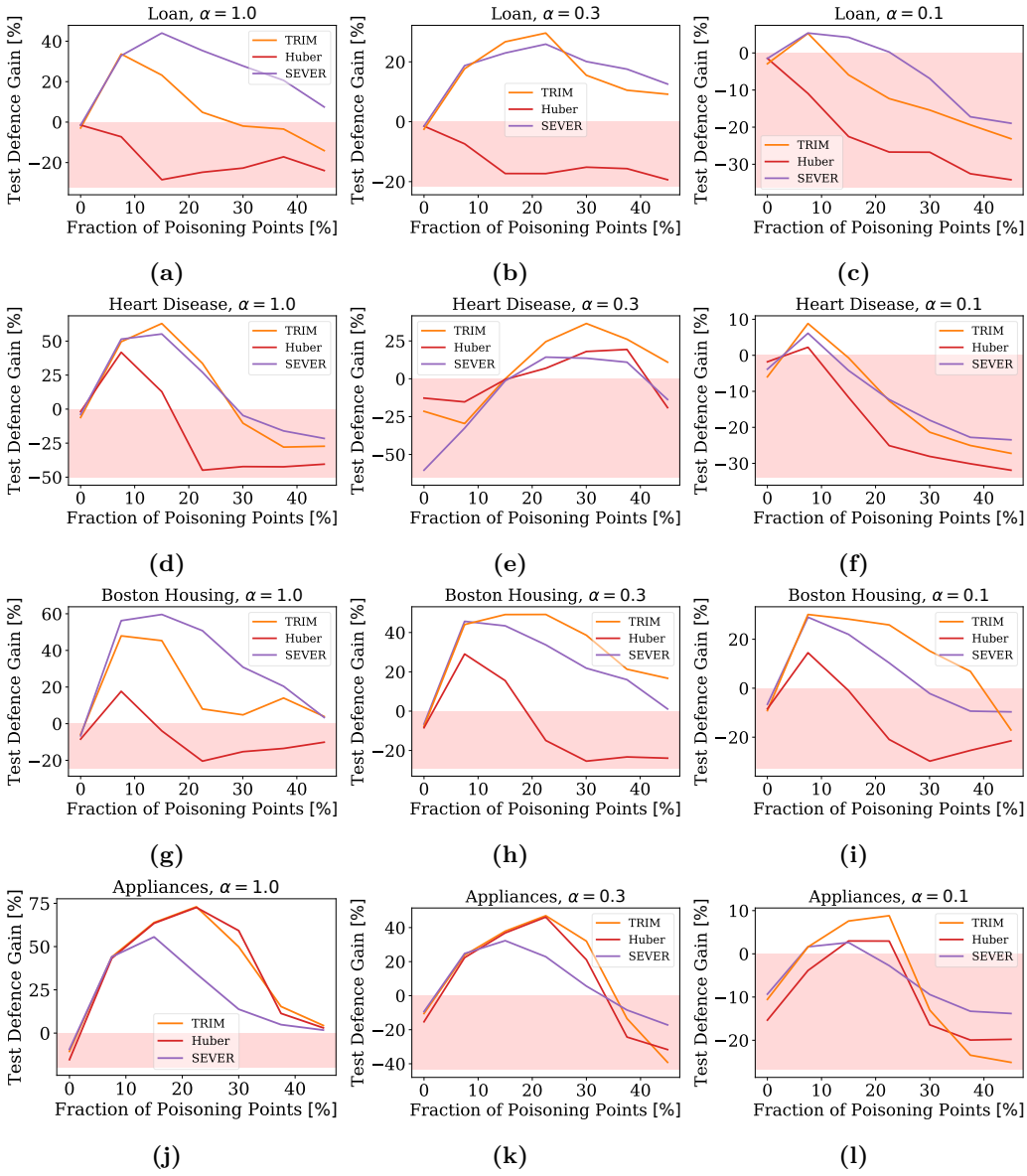


Figure 5.8: Test defence gain of linear regression when using TRIM, Huber, and SEVER. The first, second, third, and fourth rows correspond to Loan, Heart Disease, Boston Housing, and Appliances, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly.

Noisy Datasets

In this subsection, we aim to empirically support the theoretical findings presented in Ch. 4, Sect. 4.4: noisy datasets are more difficult to defend against poisoning attacks, because the feasible poisoning error increases with the noise level. Intuitively, the attacker can "hide" in the noise. We shall focus on the Boston Housing dataset. To sustain our hypothesis, we generate two additional datasets from the original Boston Housing one, but adding noise to the labels. In particular, we add zero-mean Gaussian noise to their labels,

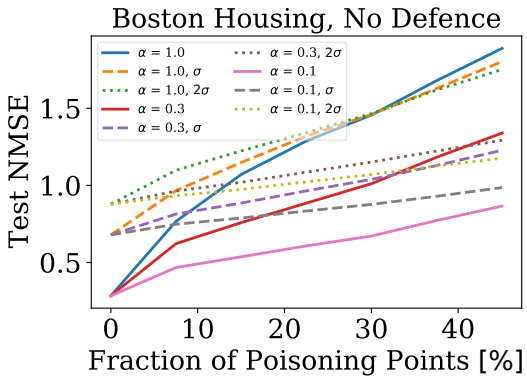


Figure 5.9: Test NMSE of linear regression on Boston Housing datasets with label noise with different standard deviations (standard Boston Housing, σ, and 2σ), when there is no defence deployed, for $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$.

with standard deviation of value σ and 2σ, respectively, where σ is the standard deviation of the original labels.

Fig. 5.9 depicts the results corresponding to the case where defences have not been deployed. Fig. 5.9 suggests that the noisy datasets present a larger NMSE for low ratios of poisoning points, and their NMSE approaches the error of the non-noisy dataset when the number of poisoning points increases. This is a consequence of the optimisation process of the poisoning points: given a maximum error achievable by the optimal poisoning points, the poisoning perturbations needed to achieve such error vary according to the values of the whole training set. Thus, the attacker needs to perturb more points in the case of the non-noisy dataset to achieve an error similar to the one for the noisy datasets.

The results in Fig. 5.10 evidence the ease of poisoning noisy datasets, according to the definition and discussion presented in Ch. 4, Sect. 4.4. Especially, when the attacker considers detectability ($\alpha = 0.3$ and $\alpha = 0.1$), the test defence gain is practically negative for the three defences. This effectively shows the important role that the characteristics of the datasets plays when considering attacks and defences.

Deep Neural Networks

We also show the scalability of our attack by targeting feed-forward DNNs. To our knowledge, this is the first study that analyses optimal data poisoning attacks targeting DNNs in regression settings. As in Ch. 3, we apply different initial random parameters $\mathbf{w}^{(0)}$ for every update of the poisoning points. This can be interpreted as assembling different randomly initialised DNNs to improve the generalisation of the poisoning points across

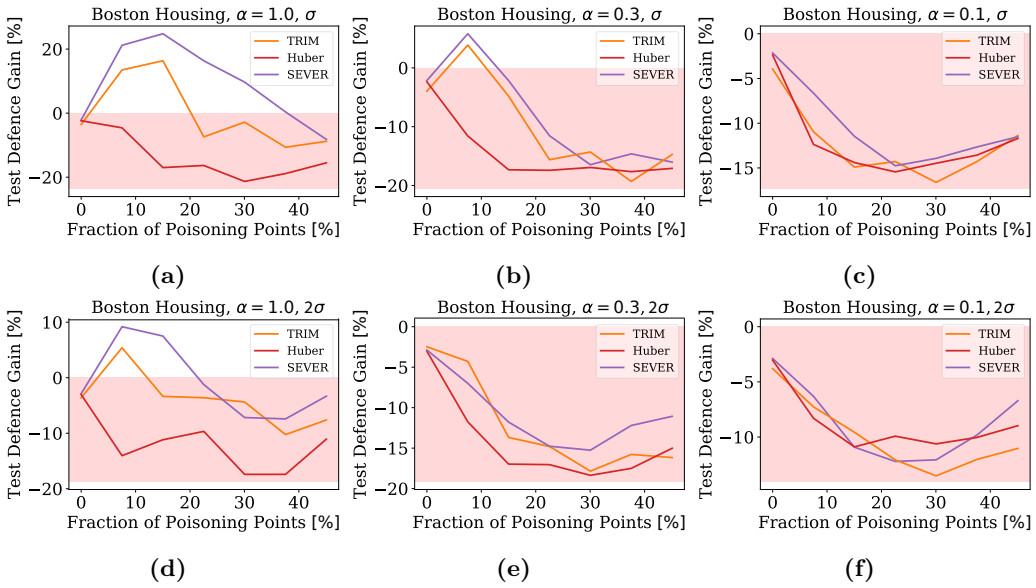


Figure 5.10: Test defence gain of linear regression on Boston Housing dataset, for different levels of label noise, when using when using TRIM, Huber, and SEVER. The first and second rows correspond to σ and 2σ, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly.

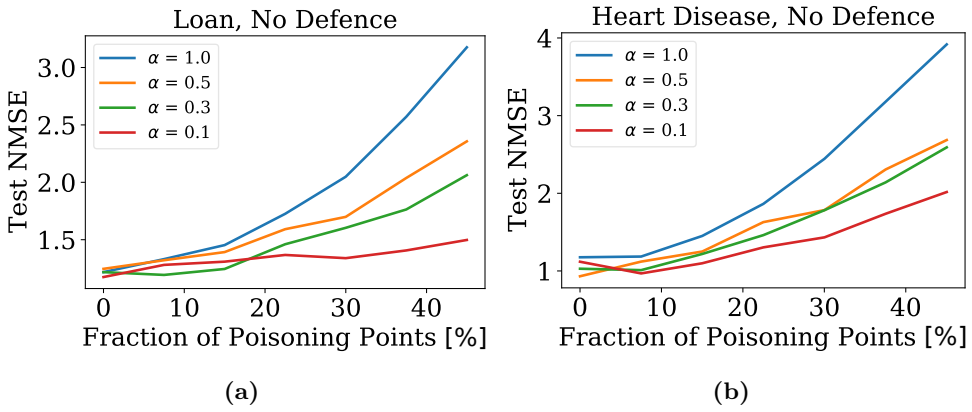


Figure 5.11: Test NMSE of the DNNs when there is no defence deployed, for $\alpha = 1$, $\alpha = 0.5$, $\alpha = 0.3$, and $\alpha = 0.1$. (a) Loan. (b) Heart Disease.

different parameter initialisations.

The case where there is no defence is represented in Fig. 5.11. The results are similar to the ones in Fig. 5.7: the test NMSE increases with α, which provides flexibility to generate a smooth transition between inliers and outliers. Interestingly, the test NMSE is larger compared to linear regression, suggesting that the DNN is more vulnerable to the poisoning attack.

Fig. 5.12 illustrates the performance of the attack against DNNs when considering TRIM, Huber, and SEVER, in a manner similar to the experiments shown for linear

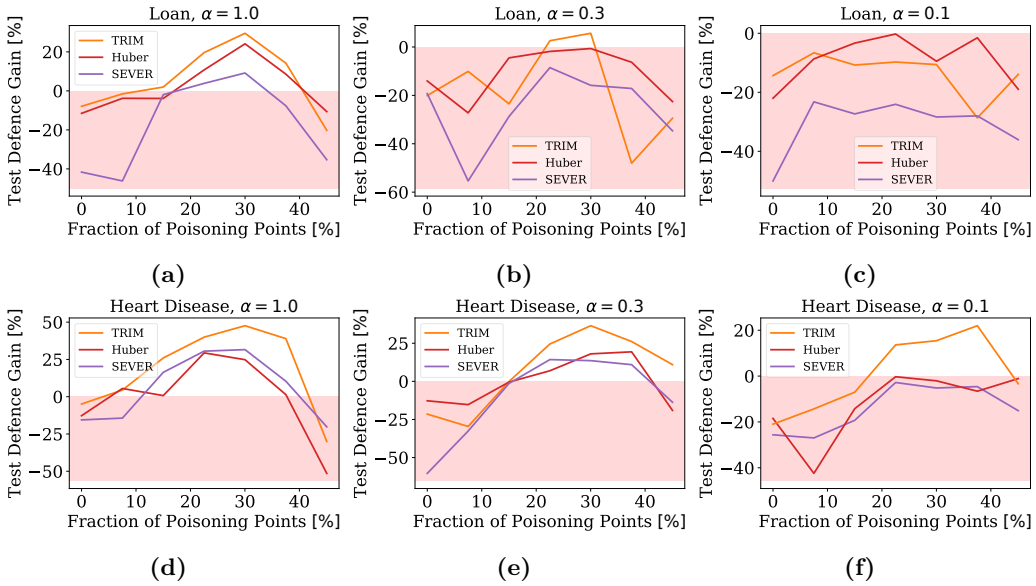


Figure 5.12: Test defence gain of the DNNs when using TRIM, Huber, and SEVER. The first and second rows correspond to Loan and Heart Disease, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly.

regression. The takeaway is that in the case of DNNs as well, most defences fall short when the attacker considers detectability. For the Loan dataset, comparing Fig. 5.12 and Fig. 5.8 suggests that, overall, the defences perform worse in the case of the DNNs, since the test defence gain is lower in the latter case. This can also be attributed to the fact that the number of features of Loan is larger, which can make it less robust to the attack.

5.5.3 BayesClean

In this subsection we evaluate our defence proposed, BayesClean, comparing it to state-of-the-art defences. Note that, unlike other defences, DUTI assumes the presence of a small clean trusted set. We highlight this by representing it differently (with a dashed line).

Linear Regression

Fig. 5.13 show the performance of BayesClean compared to the other state-of-the-art defences, when testing our attack against linear regression. In most cases, TRIM, SEVER, and Proda present a similar pattern: they perform well for low ratios of poisoning, and they worsen from a certain ratio. As mentioned previously, our hypothesis is that there is a certain number of poisoning points from which the model is too much shifted towards the poisoning distribution to be considered useful for the defences. We shall explore this

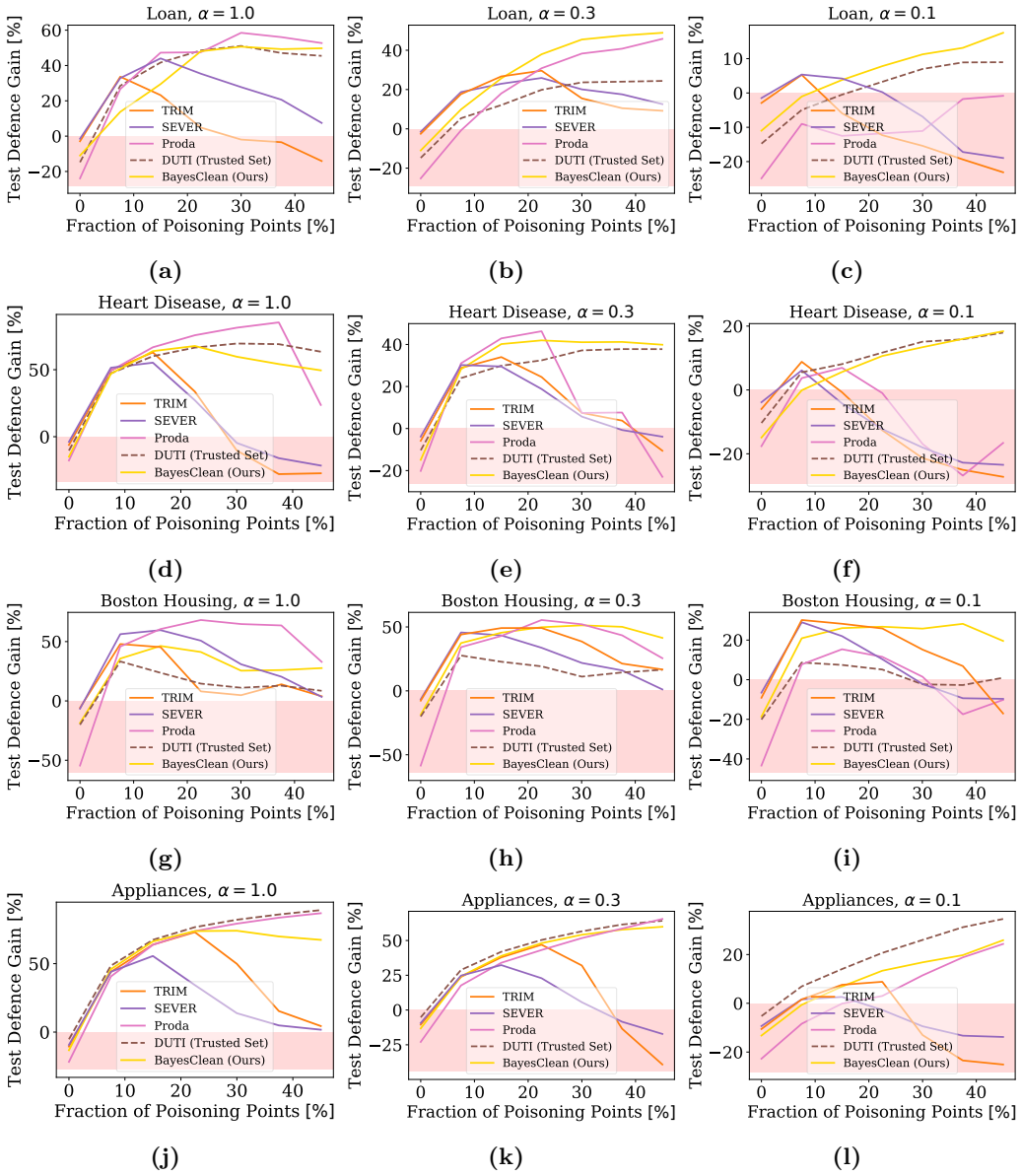


Figure 5.13: Test defence gain of linear regression when using BayesClean, compared to state-of-the-art defences. The first, second, third, and fourth rows correspond to Loan, Heart Disease, Boston Housing, and Appliances, respectively. The first, second, and third column correspond to $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, correspondingly.

hypothesis in Ch. 7. For instance, the error and/or the norm of the gradients of the poisoning points can be lower than error of the clean points on the poisoned model under those circumstances.

On the other hand, the test defence gain of BayesClean is more stable and sometimes increases with the number of poisoning points. In other words, BayesClean is competitive with the other defences for low ratios of poisoning points and aggressive attacks ($\alpha = 1$). Moreover, BayesClean improves upon most defences when the number of poisoning points

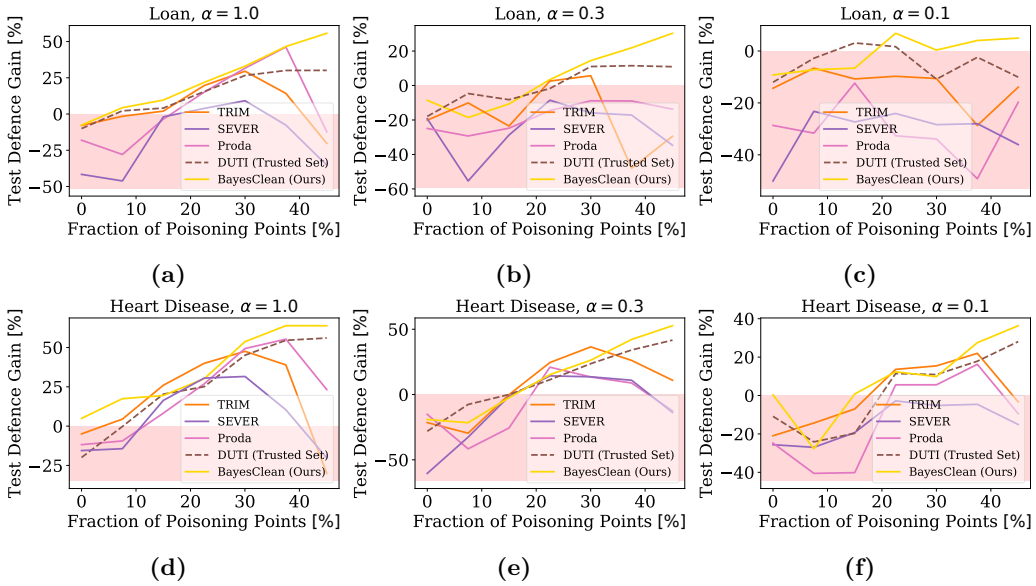


Figure 5.14: Test defence gain of the DNNs when using BayesClean, compared to state-of-the-art defences. The first row represents the results for Loan, whereas the second row corresponds to the results for Heart Disease. On the other hand, the first, second, and third columns correspond to the results for $\alpha = 1$, $\alpha = 0.3$, and $\alpha = 0.1$, respectively.

is large and when the attacker considers detectability. DUTI also performs well because it is based on a trusted validation set, however it performs poorly on Boston Housing. These results show the benefits of our proposed defence and the importance of considering the uncertainty of the model in future research. As we commented in Subsect. 5.4.2, the uncertainty (or variance) of the model can be more robust than the error of the model when the number of poisoning points is large.

Deep Neural Networks

For completeness, we also show in Fig. 5.14 the performance of our defence to protect DNNs. Even though *BayesClean* is based on a Bayesian linear regression model, its predictive variance is also able to detect poisoning points targeting the DNNs. As in the case of linear regression, our defence is competitive in most cases, being particularly useful when the attacker considers detectability and when the ratio of poisoning points is large. Furthermore, the negative values of the test defence gain suggest that defending the DNNs is more challenging than the case of linear regression. These results suggest further research on ad hoc methods to defend DNNs for regression.

5.6 Conclusions

Most of previous work on data poisoning attacks against regression models has focussed on non-adaptive attacks that maximise the error of the model. These attacks can be mitigated by deploying defence mechanisms that consider the distribution of the training data.

To cover this research gap, in this chapter we have proposed a novel attack formulation based on a multiobjective bilevel optimisation problem. This problem models the two conflicting objectives of the attacker: maximising their effectiveness and minimising their risk of being detected. This formulation is more realistic than previous approaches since in many applications users will deploy a detector to reject points that are suspicious. However, solving the proposed attack formulation is challenging as it balances the two objectives within a bilevel optimisation problem. Since the objectives can be of different order of magnitude, another challenge is that such formulation requires normalisation of the objectives, which is itself complex, to be able to accurately balance the objectives and make comparisons of the hyperparameter that balances them, α, meaningful.

We have proposed a novel detectability-risk function and an effective method to solve the optimisation problem proposed, by levering RMD, as in Ch. 3, and normalisation of objectives. Our experiments in both linear regression models and DNNs on real-world datasets show that stealthy attacks may be less effective but they bypass proposed defences and are sufficient to induce significant errors in the models. Moreover, to our knowledge, we are the first to analyse optimal data poisoning attacks targeting DNNs in regression settings. In summary, defences are in some cases not worth investing in as the resulting model has poorer performance than when no defence is employed. Hence, the deployment of defences may induce a false sense of security. Most fail against stealthy attacks but some also fail in the context in which they are claimed to work, e.g., for large ratios of poisoning points (e.g., more than 25% in some cases).

Our investigation has also evidenced the importance of the fitness of the dataset to the model as being fundamental to the ease of poisoning the model itself. In particular, we have empirically shown that noisy datasets can be more difficult to defend, which evidences the theoretical findings of Ch. 4, Sect. 4.4.

We have also proposed a novel defence against poisoning attacks in regression settings called BayesClean. Fundamentally, this defence builds upon a Bayesian linear regression

model whose hyperparameters are learnt from the training set. Then, points are rejected depending on their relative location within the predictive variance of the model. Our experiments show the effectiveness of the proposed defence and its advantage w.r.t. other defences when the ratio of poisoning points is large and when the attacker considers detectability.

Despite their importance in many practical problems, we are still in the early stages of understanding the attack surface of regression models to poisoning attacks and developing robust techniques for their defence. Numerous issues need to be investigated and challenges to be addressed. We expect that the work presented here, will enable more realistic and systematic testing of the robustness to poisoning of ML models and the development of new and more robust defences.

To understand the challenges and limitations of the poisoning attack formulations proposed in this chapter, in Ch. 6 we shall theoretically analyse some fundamental properties of the attacker's objective functions proposed in this chapter.

Chapter 6

Analysis of Optimal Indiscriminate Attacks Against Ridge Regression

Synopsis

We characterise the attacker's objective functions proposed in Ch. 5. We show that, without considering detectability constraints, the attacker's objective is convex in the poisoning labels, but non-convex in the poisoning features (or basis functions if used). We conjecture that most of the critical points (namely stationary points) of the attacker's objective are saddle points. We theoretically determine that regularising the model can prevent the attacker from being unbounded, which is coherent with the results obtained in Ch. 3. We also show the equivalence between the optimal indiscriminate attack and concave quadratic programming, if only the poisoning labels are optimised. When considering detectability constraints, we show that the attacker's objective's convexity can be brittle: because the attacker's objective is a sum of a convex objective—effectiveness—and a concave objective—detectability—the objective function can undesirably shift from convexity to concavity. These theoretical findings justify the normalisation procedure and the detectability objective designed in Ch. 5.

6.1 Introduction

In Ch. 5 we proposed a novel attack formulation, based on a multiobjective bilevel optimisation problem, to model different degrees of detectability constraints for the

attacker. We faced a series of challenges when solving this problem: the two objectives can be of different order of magnitude, which can make the optimisation process difficult since one of the objectives typically dominates the other. For this reason, we proposed a method to normalise the objectives. Moreover, if the detectability-risk function is not carefully designed, the effectiveness and detectability objectives can be redundant, which also hinders the transition from inliers to outliers. Accordingly, we designed the detectability-risk function as a product of two loss functions that penalise points that are near the bounds of a certain confidence interval. Another challenge found is that the optimisation process of the poisoning points is slow if their corresponding hypergradients are not normalised. A possible explanation is that the attacker's objective function is highly non-convex. We will investigate these issues and other properties of the attacker's objective function in this chapter.

Hence, we shall answer the following research question (Ch. 1):

- **RQ8**: Can we have a thorough theoretical understanding of optimal indiscriminate attacks against linear regression models?

In this chapter, we shall characterise the attacker's objective function in optimal indiscriminate attacks against ridge regression, i.e., L_2-regularised linear regression. This will include the analysis of its convexity, asymptotic behaviour, critical points,[1] combinations of poisoning labels, equivalent formulations, influence of soft detectability constraints, and challenges in the optimisation process. The motivation is to help design and have a better control over objective functions according to the given specifications—e.g., diversity of poisoning points—and also to illustrate the challenges and possible solutions that can improve the optimisation process.

We show that, without considering soft detectability constraints, the attacker's objective is convex in the poisoning labels, but non-convex in the poisoning basis functions. We also conjecture that most of the critical points of the attacker's objective are saddle points. This requires heuristics, e.g., normalisation of hypergradients, to help escape the saddle points and accelerate the optimisation process. With regards to the asymptotic behaviour of the attacker's objective, we theoretically determine the intuitive result that the attacker

[1] A critical point of a function is a point in its domain where all the partial derivatives of the function are zero or undefined. If all the partial derivatives at a critical point are zero, then the critical point corresponds to a stationary point.

is generally unbounded if the poisoning basis functions and/or labels are unbounded. However, regularising the model can prevent the attacker from being unbounded, which is coherent with the results obtained in classification settings in Ch. 3. We also show the equivalence between the optimal indiscriminate attack and concave quadratic programming (CQP) (Chinchuluun et al., 2005), if only the poisoning labels are optimised.

When considering soft detectability constraints, we show that the attacker's objective's convexity can be brittle: because the attacker's objective is a sum of a convex objective—effectiveness—and a negative convex (concave) objective—detectability—the objective function can undesirably transition from convexity to concavity, depending on the values of their respective Hessians and the hyperparameter that controls their trade-off, α.

Finally, we develop the attacker's KKT conditions (Karush, 1939; Kuhn and Tucker, 1951), which enables us to discuss the different solutions depending on the convexity of the attacker's objective. We also justify the detectability-risk function proposed in Ch. 5 based on a product of two convex functions, which can provide more diversity of poisoning points—and potentially increasing the difficulty to detect them. If the effectiveness objective is convex and quadratic, and the detectability-risk function is non-negative, then a quartic detectability-risk function can provide diversity for the optimal poisoning points.

In the following sections, we shall consider the optimisation of the whole poisoning set at the same time. This contrasts with most of the literature in optimal poisoning attacks, which usually focusses on the optimisation of a single poisoning point at a time.

6.2 Analysis of the Convexity of the Attacker's Objective

We shall show that the model's parameters are jointly convex[2] in the poisoning labels in the particular case where the attacker only optimises the poisoning labels, but non-convex (and non-concave) when the attacker optimises the poisoning design matrix or both the poisoning design matrix and the poisoning labels simultaneously. This is a direct consequence of the structure of the parameters in linear regression (normal equation). If the model's parameters are jointly convex in the poisoning labels, we shall see that standard attacker's

[2]A function $f(\mathbf{x}, \mathbf{y}) : \mathbb{R}^{n+m} \to \mathbb{R}$ is jointly convex in $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ if and only if it is (i) marginally convex in both $\mathbf{x}$ and $\mathbf{y}$, and (ii) convex when considering the vector $[\mathbf{x}\ \mathbf{y}]^\mathsf{T}$. In the context of mathematical optimisation, convexity usually refers to joint convexity. Note that joint convexity implies marginal convexity in both $\mathbf{x}$ and $\mathbf{y}$, but the opposite is not necessarily true (Boyd and Vandenberghe, 2004). For instance, $f(x, y) = xy$, where $x, y \in \mathbb{R}$, is marginally convex in both x and y, but not jointly convex in $[x\ y]^\mathsf{T}$.

objective functions such as the machine teaching objective or the indiscriminate attack objective (both without detectability constraints) are convex as well.

As in Ch. 5, given the input space $\mathcal{X} \subseteq \mathbb{R}^m$ and the continuous label space, $\mathcal{Y} \subseteq \mathbb{R}$, the learner aims to estimate the mapping $f : \mathcal{X} \to \mathcal{Y}$. Given the clean training set $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}_{\text{tr}_i}, y_{\text{tr}_i})\}_{i=1}^{n_{\text{tr}}} = (\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ with n_{tr} IID samples drawn from the underlying probability distribution $p(\mathcal{X}, \mathcal{Y})$, we can estimate f with a model $\mathcal{M}$ trained by minimising an objective function $\mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w})$ w.r.t. its parameters, $\mathbf{w} \in \mathbb{R}^d$. The parameters learnt under the clean training set, $\mathcal{D}_{\text{tr}}$, are therefore $\mathbf{w}^\star_{\text{tr}}$.

In a poisoning attack, the adversary aims to inject a set of n_{p} poisoning data points, $\mathcal{D}_{\text{p}} = \{(\mathbf{x}_{\text{p}_k}, y_{\text{p}_k})\}_{k=1}^{n_{\text{p}}} = (\mathbf{X}_{\text{p}}, \mathbf{y}_{\text{p}})$, in the training set. Thus, under the poisoning attack, the learner minimises the objective function $\mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}, \mathbf{w})$. The parameters learnt under the poisoned training set, $\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}$, are thus $\mathbf{w}^\star$.

Note that the following derivation is also applicable to non-linear parametric regression, as long as the model is linear in the parameters, e.g., polynomial regression, logarithmic regression or exponential regression. We therefore extend the class of models by considering linear combinations of $d-1$ fixed non-linear functions of the input: $y = \boldsymbol{\phi}_e(\mathbf{x})^\mathsf{T}\mathbf{w}$, where $\boldsymbol{\phi}_e(\mathbf{x}) \in \mathbb{R}^d = [1 \ \phi_1(\mathbf{x}) \ \dots \ \phi_{d-1}(\mathbf{x})]^\mathsf{T}$ are known as basis functions.[3] Thus, the total number of parameters in this model is d, including the bias term. By using non-linear basis functions,[4] we allow the function y to be a non-linear function of the input vector $\mathbf{x}$.[5] Functions of this form are called, however, linear models because this function is linear in the parameters. We define the (clean) training design matrix $\boldsymbol{\Phi}_{\text{tr}} = \left[\boldsymbol{\phi}(\mathbf{x}_{\text{tr}_1}) \ \dots \ \boldsymbol{\phi}(\mathbf{x}_{\text{tr}_{n_{\text{tr}}}})\right]^\mathsf{T} = \left[\boldsymbol{\phi}_{\text{tr}_1} \ \dots \ \boldsymbol{\phi}_{\text{tr}_{n_{\text{tr}}}}\right]^\mathsf{T}$, validation design matrix $\boldsymbol{\Phi}_{\text{val}} = \left[\boldsymbol{\phi}(\mathbf{x}_{\text{val}_1}) \ \dots \ \boldsymbol{\phi}(\mathbf{x}_{\text{val}_{n_{\text{val}}}})\right]^\mathsf{T} = \left[\boldsymbol{\phi}_{\text{val}_1} \ \dots \ \boldsymbol{\phi}_{\text{val}_{n_{\text{val}}}}\right]^\mathsf{T}$, and the poisoning design matrix $\boldsymbol{\Phi}_{\text{p}} = \left[\boldsymbol{\phi}(\mathbf{x}_{\text{p}_1}) \ \dots \ \boldsymbol{\phi}(\mathbf{x}_{\text{p}_{n_{\text{p}}}})\right]^\mathsf{T} = \left[\boldsymbol{\phi}_{\text{p}_1} \ \dots \ \boldsymbol{\phi}_{\text{p}_{n_{\text{p}}}}\right]^\mathsf{T}$, and their corresponding extended versions $\boldsymbol{\Phi}_{\text{tr}_e} = \left[\boldsymbol{\phi}_{\text{tr}_{e1}} \ \dots \ \boldsymbol{\phi}_{\text{tr}_{e\,n_{\text{tr}}}}\right]^\mathsf{T}$, $\boldsymbol{\Phi}_{\text{val}_e} = \left[\boldsymbol{\phi}_{\text{val}_{e1}} \ \dots \ \boldsymbol{\phi}_{\text{val}_{e\,n_{\text{val}}}}\right]^\mathsf{T}$, and $\boldsymbol{\Phi}_{\text{p}_e} = \left[\boldsymbol{\phi}_{\text{p}_{e_1}} \ \dots \ \boldsymbol{\phi}_{\text{p}_{e\,n_{\text{p}}}}\right]^\mathsf{T}$.

For simplicity, we assume that the attacker can manipulate the set of basis functions $\boldsymbol{\phi}$ directly, which trivially reduce to the features when $\boldsymbol{\phi} = \mathbf{x}$. Note, however, that the con-

[3] To compact our notation, we define $\boldsymbol{\phi}_e(\boldsymbol{x}) = [1 \ \boldsymbol{\phi}(\boldsymbol{x})]^\mathsf{T}$ as the extended vector of basis functions, so that $\boldsymbol{\phi}_e(\boldsymbol{x})^\mathsf{T}\mathbf{w} = w_0 + \boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}\mathbf{w}_{1:(d-1)}$, where $\mathbf{w} = [w_0 \ \mathbf{w}_{1:(d-1)}]^\mathsf{T}$ and w_0 is the bias term.

[4] This also include linear basis functions, such as the identity function $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$.

[5] Indeed, the choice of a particular basis function can affect the performance of the attack, and the attacker may not be aware of the defender's basis function. The study of the robustness of basis functions to the attack is left as future work.

vexity of the attacker's objective function depends on the basis functions themselves, which are functions of the features of the data: $\boldsymbol{\phi}(\mathbf{x})$. Intuitively, because of this transformation, the objective function is not convex in $\mathbf{X}_\mathrm{p}$ in general.

Let us recall that the solution to the L_2-regularised least-squares problem is:

$$\begin{aligned}\mathbf{w}^\star(\boldsymbol{\Phi}_\mathrm{p}, \mathbf{y}_\mathrm{p}) &= \left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\boldsymbol{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}\boldsymbol{\Phi}_{\mathrm{p}_e}\right)^{-1}\left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_\mathrm{p}\right)\\ &= \mathbf{S}_\lambda(\boldsymbol{\Phi}_\mathrm{p})^{-1}\left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_\mathrm{p}\right),\end{aligned} \tag{6.1}$$

where the regularised scatter matrix of the complete training set $\mathbf{S}_\lambda(\boldsymbol{\Phi}_\mathrm{p}) = \boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\boldsymbol{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}\boldsymbol{\Phi}_{\mathrm{p}_e}$ is assumed to be positive definite. In (6.1), we define the block matrix $\mathbf{I}_{d_e} = \mathbf{I}_d - \mathbf{e}_1\mathbf{e}_1^\mathsf{T} = \begin{bmatrix} 0 & \mathbf{0}_{d-1}^\mathsf{T} \\ \mathbf{0}_{d-1} & \mathbf{I}_{d-1}\end{bmatrix}$, where $\mathbf{I}_{d-1} \in \mathbb{R}^{(d-1)\times(d-1)}$ is the identity matrix, $\mathbf{0}_{d-1} \in \mathbb{R}^{d-1}$ is a column vector of zeros, and $\mathbf{e}_1 = [1\ \mathbf{0}_{d-1}^\mathsf{T}]^\mathsf{T}$ is a basis vector.

6.2.1 Convexity in the Poisoning Labels

If the attacker modifies only the label of the poisoning points, each of model's parameters is an affine mapping of $\mathbf{y}_\mathrm{p}$ and hence both jointly convex and jointly concave in $\mathbf{y}_\mathrm{p}$. This can be seen through the zeroth-order necessary and sufficient condition for convexity:

$$\begin{aligned}\mathbf{w}^\star(\tau\mathbf{y}_\mathrm{p} + (1-\tau)\mathbf{y}'_\mathrm{p}) &= \mathbf{S}_\lambda^{-1}\left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}(\tau\mathbf{y}_\mathrm{p} + (1-\tau)\mathbf{y}'_\mathrm{p})\right)\\ &= \tau\mathbf{S}_\lambda^{-1}\left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_\mathrm{p}\right) + (1-\tau)\mathbf{S}_\lambda^{-1}\left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}'_\mathrm{p}\right),\end{aligned} \tag{6.2}$$

where $\tau \in [0,1]$ and $\mathbf{y}'_\mathrm{p} \neq \mathbf{y}_\mathrm{p}$. In the literature, most of the attacker's objectives typically involve the model's parameters in some of these forms: $-\mathbf{a}^\mathsf{T}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})$, $||\mathbf{A}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_p$ $(p \geq 1)$, or $||\mathbf{A}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_p^p$ ($p \geq 2$ and p even), where $\mathbf{a} \in \mathbb{R}^d$ and $\mathbf{A} \in \mathbb{R}^{d\times d}$. Let us assume $\mathbf{a}$ and $\mathbf{A}$ do not depend on $\mathbf{y}_\mathrm{p}$. Since the operation $-\mathbf{a}^\mathsf{T}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})$ is an affine mapping, and each of the model's parameters is an affine mapping of $\mathbf{y}_\mathrm{p}$, $-\mathbf{a}^\mathsf{T}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})$ is also an affine mapping of $\mathbf{y}_\mathrm{p}$ and thus both convex and concave. This means that adding $-\mathbf{a}^\mathsf{T}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})$ to a function

preserves the convexity of the function. Regarding $||\mathbf{A}\mathbf{w}^\star||_p$:

$$\begin{aligned}
||\mathbf{A}\mathbf{w}^\star(\tau\mathbf{y}_\mathrm{p}+(1-\tau)\mathbf{y}'_\mathrm{p})||_p &= \left|\left|\mathbf{A}\mathbf{S}_\lambda^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr}+\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}(\tau\mathbf{y}_\mathrm{p}+(1-\tau)\mathbf{y}'_\mathrm{p})\right)\right|\right|_p \\
&= \Big|\Big|\tau\mathbf{A}\mathbf{S}_\lambda^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr}+\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_\mathrm{p}\right) \\
&\quad +(1-\tau)\mathbf{A}\mathbf{S}_\lambda^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr}+\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}'_\mathrm{p}\right)\Big|\Big|_p, \qquad (6.3)\\
&\leq \tau\left|\left|\mathbf{A}\mathbf{S}_\lambda^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr}+\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_\mathrm{p}\right)\right|\right|_p \\
&\quad +(1-\tau)\left|\left|\mathbf{A}\mathbf{S}_\lambda^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr}+\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}'_\mathrm{p}\right)\right|\right|_p,
\end{aligned}$$

due to the triangle inequality, which shows that $||\mathbf{A}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_p$ is jointly convex in $\mathbf{y}_\mathrm{p}$. This can also be seen by combining the affine mapping property and the convexity of the p-norm of a vector. Finally, for $||\mathbf{A}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_p^p$ ($p \geq 2$ and p even), given that $||\mathbf{A}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_p$ is convex and non-negative, $||\mathbf{A}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_p^p$ is convex as well. As an example of joint convex attacker's objective function in $\mathbf{y}_\mathrm{p}$, the machine teaching objective without detectability constraints $\mathcal{A}(\mathbf{y}_\mathrm{p}) = \frac{1}{2d}||\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) - \mathbf{w}^\star_\mathrm{target}||_2^2 = \frac{1}{2d}||\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2 - \frac{1}{d}\mathbf{w}^\mathsf{T}_\mathrm{target}\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) + \frac{1}{2d}||\mathbf{w}^\star_\mathrm{target}(\mathbf{y}_\mathrm{p})||_2^2$ is jointly convex in $\mathbf{y}_\mathrm{p}$ because a non-negative weighted sum of convex functions is also convex.[6] Similarly, the indiscriminate attack objective without detectability constraints $\mathcal{A}(\mathbf{y}_\mathrm{p}) = \frac{1}{2n_\mathrm{val}}||\mathbf{y}_\mathrm{val} - \mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2 = \frac{1}{2n_\mathrm{val}}||\mathbf{y}_\mathrm{val}||_2^2 - \frac{1}{n_\mathrm{val}}\mathbf{y}^\mathsf{T}_\mathrm{val}\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) + \frac{1}{2n_\mathrm{val}}||\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2$ is also jointly convex in $\mathbf{y}_\mathrm{p}$.[7]

Thus, we can conclude that the attacker's objective is convex in the poisoning labels. Next, we will analyse the convexity in the basis functions of the poisoning points.

6.2.2 Convexity in the Basis Functions

If the attacker modifies the basis functions of the poisoning points (considering that their labels are fixed), the model's parameters are not jointly convex or jointly concave in $\mathbf{\Phi}_\mathrm{p}$ in general. We shall see this through the zeroth and the second-order conditions for convexity.

[6] Note that adding a constant b to a function preserves the convexity of the function.
[7] Recall that "val" refers to the validation set.

Zeroth-Order Condition for Convexity

By expanding the zeroth-order condition for convexity, $\exists \tau, \mathbf{\Phi}_{\mathrm{p}}, \mathbf{\Phi}'_{\mathrm{p}} \neq \mathbf{\Phi}_{\mathrm{p}}$:

$$\begin{aligned}
\mathbf{w}^{\star}(\tau\mathbf{\Phi}_{\mathrm{p}} + (1-\tau)\mathbf{\Phi}'_{\mathrm{p}}) &= \left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + (\tau\mathbf{\Phi}_{\mathrm{p}_e} + (1-\tau)\mathbf{\Phi}'_{\mathrm{p}_e})^{\mathsf{T}}(\tau\mathbf{\Phi}_{\mathrm{p}_e} + (1-\tau)\mathbf{\Phi}'_{\mathrm{p}_e})\right)^{-1} \\
&\quad \left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{tr}} + (\tau\mathbf{\Phi}_{\mathrm{p}_e} + (1-\tau)\mathbf{\Phi}'_{\mathrm{p}_e})^{\mathsf{T}}\mathbf{y}_{\mathrm{p}}\right) \\
&= \left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + \tau^2\mathbf{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\mathbf{\Phi}'_{\mathrm{p}_e} + \tau(1-\tau)\left(\mathbf{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\mathbf{\Phi}'_{\mathrm{p}_e} + \mathbf{\Phi}_{\mathrm{p}_e}^{\prime\mathsf{T}}\mathbf{\Phi}_{\mathrm{p}_e}\right)\right. \\
&\quad \left.+(1-\tau)^2\mathbf{\Phi}_{\mathrm{p}_e}^{\prime\mathsf{T}}\mathbf{\Phi}'_{\mathrm{p}_e}\right)^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{tr}} + \tau\mathbf{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{p}} + (1-\tau)\mathbf{\Phi}_{\mathrm{p}_e}^{\prime\mathsf{T}}\mathbf{y}_{\mathrm{p}}\right) \\
&\not\preceq \tau\left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + \mathbf{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\mathbf{\Phi}_{\mathrm{p}_e}\right)^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{tr}} + \mathbf{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{p}}\right) \\
&\quad + (1-\tau)\left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + \mathbf{\Phi}_{\mathrm{p}_e}^{\prime\mathsf{T}}\mathbf{\Phi}'_{\mathrm{p}_e}\right)^{-1}\left(\mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{tr}} + \mathbf{\Phi}_{\mathrm{p}_e}^{\prime\mathsf{T}}\mathbf{y}_{\mathrm{p}}\right).
\end{aligned} \tag{6.4}$$

In this case, the inequality, denoted by the element-wise partial order of vectors associated with the non-negative orthant,[8] is made between the elements of the vectors on each side of the inequality. If $\mathbf{a}, \mathbf{b}$ are two vectors in $\mathbb{R}^n$, the notation $u \preceq v$ corresponds to the element-wise inequality, that is: $\mathbf{a} \preceq \mathbf{b} \iff a_i \leq b_i,\ i = 1, \ldots, n$. In words, all the elements of the vector $\mathbf{b} - \mathbf{a}$ are non-negative. Conversely, $\mathbf{a} \not\preceq \mathbf{b} \iff \exists a_i : a_i > b_i$. That is, at least one element of the vector $\mathbf{b} - \mathbf{a}$ is negative.

There exist uncountable examples that satisfy the condition (6.4), and they can be easily found. For instance, for reproducibility purposes, we set $\tau = 0.5$ and $\lambda = 0.1$, and we generate $\mathbf{\Phi}_{\mathrm{tr}} \in \mathbb{R}^{6\times 2}, \phi_{\mathrm{tr}_{i,j}} \sim \mathcal{U}(-0.50, 0.50)$; $\mathbf{y}_{\mathrm{tr}} = \mathbf{\Phi}_{\mathrm{tr}_e}\mathbf{w}_{\mathrm{true}} + \boldsymbol{\epsilon}$, $[\mathbf{w}_{\mathrm{true}}]_k \sim \mathcal{N}(0.00, 0.50^2)$, $[\boldsymbol{\epsilon}]_i \sim \mathcal{N}(0.00, 0.05^2)$, $\mathbf{\Phi}_{\mathrm{p}} \in \mathbb{R}^{3\times 2}, [\mathbf{\Phi}_{\mathrm{p}}]_{i,j} \sim \mathcal{U}(-1.00, 1.00)$; $\mathbf{\Phi}'_{\mathrm{p}} \in \mathbb{R}^{3\times 2}, [\mathbf{\Phi}'_{\mathrm{p}}]_{i,j} \sim \mathcal{U}(-1.00, 1.00)$; $i = 1, \ldots, 6$; $j = 1, 2$; $k = 1, 2, 3$:

[8] The non-negative orthant, which is a proper cone, is the generalisation of the first quadrant to n-dimensions. A cone $\mathcal{K} \subseteq \mathbb{R}^n$ is called a proper cone if it satisfies the following: $\mathcal{K}$ is convex; $\mathcal{K}$ is closed; $\mathcal{K}$ is solid, which means it has non-empty interior; and $\mathcal{K}$ is pointed, which means that it contains no line.

$$\boldsymbol{\Phi}_{\mathrm{tr}_e} = \begin{bmatrix} 1.00 & 0.29 & 0.35 \\ 1.00 & 0.09 & 0.12 \\ 1.00 & 0.41 & 0.11 \\ 1.00 & 0.25 & -0.49 \\ 1.00 & 0.31 & 0.06 \\ 1.00 & 0.23 & 0.35 \end{bmatrix}, \quad \mathbf{w}_{\mathrm{true}} = \begin{bmatrix} 0.17 \\ -0.45 \\ -0.09 \end{bmatrix}, \quad \boldsymbol{\epsilon} = 10^{-2} \begin{bmatrix} -0.80 \\ -1.87 \\ -8.56 \\ 2.84 \\ -0.51 \\ 8.50 \end{bmatrix},$$

$$\mathbf{y}_{\mathrm{tr}} = \begin{bmatrix} 0.00 \\ 0.10 \\ -0.11 \\ 0.13 \\ 0.02 \\ 0.12 \end{bmatrix}, \quad \mathbf{y}_{\mathrm{p}} = \begin{bmatrix} 0.05 \\ -0.09 \\ -0.97 \end{bmatrix}, \tag{6.5}$$

$$\boldsymbol{\Phi}_{\mathrm{p}_e} = \begin{bmatrix} 1.00 & -0.46 & 0.29 \\ 1.00 & 0.60 & 0.39 \\ 1.00 & -0.59 & -0.81 \end{bmatrix}, \quad \boldsymbol{\Phi}'_{\mathrm{p}_e} = \begin{bmatrix} 1.00 & 0.48 & -0.59 \\ 1.00 & 0.95 & -0.65 \\ 1.00 & 0.51 & 0.82 \end{bmatrix},$$

Under the previous values, inequality (6.4) becomes:

$$\begin{bmatrix} -0.18 < 0.04 \\ 0.47 > -0.23 \\ -0.07 < -0.06 \end{bmatrix}. \tag{6.6}$$

However, if we change the values of the poisoning points:

$$\boldsymbol{\Phi}_{\mathrm{p}_e} = \begin{bmatrix} 1.00 & -0.46 & 0.82 \\ 1.00 & -0.62 & -0.62 \\ 1.00 & -0.13 & 0.42 \end{bmatrix}, \quad \boldsymbol{\Phi}'_{\mathrm{p}_e} = \begin{bmatrix} 1.00 & 0.53 & 0.92 \\ 1.00 & -0.46 & -0.57 \\ 1.00 & -0.41 & -0.24 \end{bmatrix}, \tag{6.7}$$

the inequality results in:

$$\begin{bmatrix} -0.10 > -0.12 \\ 0.39 < 0.44 \\ -0.04 > -0.14. \end{bmatrix}. \tag{6.8}$$

Since the inequality changes for different values of the poisoning points, we can conclude that the model's parameters are not jointly convex or jointly concave in $\boldsymbol{\Phi}_{\mathrm{p}}$. To support

this previous statement, we shall examine the second-order condition for convexity, which is equivalent to the zeroth-order condition.

Second-Order Condition for Convexity

The second-order condition states that a function is convex if and only if its Hessian matrix is positive semidefinite. A necessary—but not sufficient—condition for a matrix to be positive semidefinite is that all of its diagonal entries are non-negative. The derivative of the model's parameters w.r.t. $\phi_{\mathrm{p}_{i,j}} \in \mathbf{\Phi}_\mathrm{p}$, that is, w.r.t. a particular basis function $\boldsymbol{\phi}_{\mathrm{p}_j}$ of a particular poisoning point $\boldsymbol{\phi}_{\mathrm{p}_{e_i}}$, where $i = 1, \ldots, n_\mathrm{p}$, $j = 1, \ldots, d-1$, is:

$$\begin{aligned}\frac{\partial \mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})}{\partial \phi_{\mathrm{p}_{i,j}}} &= \mathbf{S}_\lambda(\mathbf{\Phi}_\mathrm{p})^{-1}(y_{\mathrm{p}_i}\mathbf{e}_{j+1} - \mathbf{R}_{i,j+1}\mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})) \\ &= \mathbf{S}_\lambda(\mathbf{\Phi}_\mathrm{p})^{-1}\left(\left(y_{\mathrm{p}_i} - \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^\mathsf{T}\mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})\right)\mathbf{e}_{j+1} - w_j^\star(\mathbf{\Phi}_\mathrm{p})\boldsymbol{\phi}_{\mathrm{p}_{e_i}}\right),\end{aligned} \tag{6.9}$$

where $\mathbf{e}_{j+1} = [\mathbf{0}_j^\mathsf{T}\ 1\ \mathbf{0}_{d-j-1}^\mathsf{T}]^\mathsf{T}$ is a basis vector, y_{p_i} is the poisoning label of the ith poisoning point, $\mathbf{R}_{i,j+1} = \mathbf{P}_{i,j+1} + \mathbf{P}_{i,j+1}^\mathsf{T}$, $\mathbf{P}_{i,j+1} = \boldsymbol{\phi}_{\mathrm{p}_{e_i}}\mathbf{e}_{j+1}^\mathsf{T} = [\mathbf{0}_{d\times j}\ \boldsymbol{\phi}_{\mathrm{p}_{e_i}}\ \mathbf{0}_{d\times(d-j-1)}]$, and $w_j^\star$ is the jth component of the poisoned parameters $\mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p}) = [w_0^\star(\mathbf{\Phi}_\mathrm{p})\ \ldots\ w_j^\star(\mathbf{\Phi}_\mathrm{p})\ \ldots\ w_{d-1}^\star(\mathbf{\Phi}_\mathrm{p})]^\mathsf{T}$, where $w_0^\star(\mathbf{\Phi}_\mathrm{p})$ is the poisoned bias term. The corresponding second derivative, i.e., the corresponding diagonal entries of the Hessian matrix of each parameter, is:

$$\begin{aligned}\frac{\partial^2 \mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})}{\partial \phi_{\mathrm{p}_{i,j}}^2} &= 2\mathbf{S}_\lambda(\mathbf{\Phi}_\mathrm{p})^{-1}\left(\mathbf{R}_{i,j+1}\mathbf{S}_\lambda(\mathbf{\Phi}_\mathrm{p})^{-1}(\mathbf{R}_{i,j+1}\mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p}) - y_{\mathrm{p}_i}\mathbf{e}_{j+1}) - \mathbf{E}_{j+1}\mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})\right) \\ &= 2\mathbf{S}_\lambda(\mathbf{\Phi}_\mathrm{p})^{-1}\left(\mathbf{R}_{i,j+1}\mathbf{S}_\lambda(\mathbf{\Phi}_\mathrm{p})^{-1}\left(\left(\boldsymbol{\phi}_{\mathrm{p}_{e_i}}^\mathsf{T}\mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p}) - y_{\mathrm{p}_i}\right)\mathbf{e}_{j+1} + w_j^\star(\mathbf{\Phi}_\mathrm{p})\boldsymbol{\phi}_{\mathrm{p}_{e_i}}\right)\right. \\ &\quad \left. - w_j^\star(\mathbf{\Phi}_\mathrm{p})\mathbf{e}_{j+1}\right),\end{aligned} \tag{6.10}$$

where

$$\mathbf{E}_{j+1} = \mathbf{e}_{j+1}\mathbf{e}_{j+1}^\mathsf{T} = \begin{bmatrix} & \mathbf{0}_j & \\ \mathbf{0}_{d\times j} & 1 & \mathbf{0}_{d\times(d-j-1)} \\ & \mathbf{0}_{d-j-1} & \end{bmatrix} = \begin{bmatrix} & \mathbf{0}_{j\times d} & \\ \mathbf{0}_j^\mathsf{T} & 1 & \mathbf{0}_{d-j-1}^\mathsf{T} \\ & \mathbf{0}_{(d-j-1)\times d} & \end{bmatrix}. \tag{6.11}$$

From expression (6.10), $\exists \phi_{\mathrm{p}_{i,j}}, w_k^\star(\mathbf{\Phi}_\mathrm{p}) \in \mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p}) : \frac{\partial^2 w_k^\star(\mathbf{\Phi}_\mathrm{p})}{\partial \phi_{\mathrm{p}_{i,j}}^2} < 0$. Thus, the sign of the diagonal entries of the Hessian for each of the model's parameters can vary according to the values of the poisoning points, and these entries can be negative. Following the same

example as before, we use the same data and define:

$$\mathbf{\Phi}_{\mathrm{p}_e} = \begin{bmatrix} 1.00 & -0.46 & 0.29 \\ 1.00 & 0.60 & 0.39 \\ 1.00 & -0.59 & -0.81 \end{bmatrix}. \tag{6.12}$$

Let us study the case of $\phi_{\mathrm{p}_{1,1}} = -0.46$ (which corresponds to $\phi_{\mathrm{p}_{e_{1,2}}}$). In this case, we obtain $\frac{\partial^2 \mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})}{\partial \phi^2_{\mathrm{p}_{1,1}}} = [-0.29\ 1.57\ -0.90]^\mathsf{T}$. However, if we modify, for example, $\phi_{\mathrm{p}_{1,1}} = 0.48$, we obtain $\frac{\partial^2 \mathbf{w}^\star(\mathbf{\Phi}_\mathrm{p})}{\partial \phi^2_{\mathrm{p}_{1,1}}} = [0.09\ -0.80\ 0.37]^\mathsf{T}$, showing that the signs of the diagonal entries of the Hessian change and hence confirming that the model's parameters are not marginally convex or marginally concave in $\phi_{\mathrm{p}_{i,j}}$, and therefore they are not jointly convex or jointly concave in $\mathbf{\Phi}_\mathrm{p}$.

This can intuitively be seen in the following example: assume $\mathbf{w}^\star = w^\star \in \mathbb{R}$ (no bias term), unidimensional basis functions, i.e., $\boldsymbol{\phi}_{\mathrm{tr}_{e_i}} = \phi_{\mathrm{tr}_i} \in \mathbb{R}$, $i = 1, \ldots, n_\mathrm{tr}$, $\boldsymbol{\phi}_{\mathrm{p}_{e_i}} = \phi_{\mathrm{p}_i} \in \mathbb{R}$, $i = 1, \ldots, n_\mathrm{p}$, and a single poisoning point $(\phi_\mathrm{p}, y_\mathrm{p})$. Then the model's parameter reduces to:

$$w^\star(\phi_\mathrm{p}) = \frac{\sum_{i=1}^{n_\mathrm{tr}} \phi_{\mathrm{tr}_i} y_{\mathrm{tr}_i} + \phi_\mathrm{p} y_\mathrm{p}}{\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda + \phi^2_\mathrm{p}}. \tag{6.13}$$

The second-order condition for convexity reduces in our case to $\frac{\partial^2 w^\star(\phi_\mathrm{p})}{\partial \phi^2_\mathrm{p}} \geq 0$. The first derivative is:

$$\frac{\partial w^\star(\phi_\mathrm{p})}{\partial \phi_\mathrm{p}} = \frac{y_\mathrm{p}(\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda - \phi^2_\mathrm{p}) - 2\phi_\mathrm{p} \sum_{i=1}^{n_\mathrm{tr}} \phi_{\mathrm{tr}_i} y_{\mathrm{tr}_i}}{(\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda + \phi^2_\mathrm{p})^2}. \tag{6.14}$$

The second derivative is:

$$\frac{\partial^2 w^\star(\phi_\mathrm{p})}{\partial \phi^2_\mathrm{p}} = \frac{-2(\sum_{i=1}^{n_\mathrm{tr}} \phi_{\mathrm{tr}_i} y_{\mathrm{tr}_i}(\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda - 3\phi^2_\mathrm{p}) + \phi_\mathrm{p} y_\mathrm{p}(3\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + 3\lambda - \phi^2_\mathrm{p}))}{(\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda + \phi^2_\mathrm{p})^3}. \tag{6.15}$$

The denominator is always non-negative, so the condition for convexity reduces to a cubic polynomial inequality in ϕ_p:

$$\begin{aligned} p(\phi_\mathrm{p}) = 2y_\mathrm{p}\phi^3_\mathrm{p} + 6\left(\sum_{i=1}^{n_\mathrm{tr}} \phi_{\mathrm{tr}_i} y_{\mathrm{tr}_i}\right)\phi^2_\mathrm{p} - 6y_\mathrm{p}\left(\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda\right)\phi_\mathrm{p} \\ - 2\sum_{i=1}^{n_\mathrm{tr}} \phi_{\mathrm{tr}_i} y_{\mathrm{tr}_i}\left(\sum_{i=1}^{n_\mathrm{tr}} \phi^2_{\mathrm{tr}_i} + \lambda\right) \geq 0. \end{aligned} \tag{6.16}$$

By the intermediate value theorem, every polynomial of odd degree over the reals and with real coefficients must have at least one real zero. This is because $\lim_{\phi_\mathrm{p} \to \infty} p(\phi_\mathrm{p}) =$

$\text{sgn}(y_\text{p})\infty$ and $\lim_{\phi_\text{p}\to-\infty} p(\phi_\text{p}) = -\text{sgn}(y_\text{p})\infty$,[9] and polynomials are always continuous in $(-\infty, \infty)$. Since the polynomial is cubic and has real coefficients, it must take every value in the interval $\phi_\text{p} \in (-\infty, \infty)$, in particular 0. Hence, it must have at least one real root and therefore $w^\star(\phi_\text{p})$ has at least one inflection point. This shows that even in this simplified case $w^\star(\phi_\text{p})$ is not convex or concave $\forall\phi_\text{p}$.

Thus, objectives of the form $-\mathbf{a}^\top\mathbf{w}^\star(\mathbf{\Phi}_\text{p})$, $||\mathbf{A}\mathbf{w}^\star(\mathbf{\Phi}_\text{p})||_p$, or $||\mathbf{A}\mathbf{w}^\star(\mathbf{\Phi}_\text{p})||_p^p$ are also non-convex in $\mathbf{\Phi}_\text{p}$ in general, due to the properties of the convexity of composition of functions.

In conclusion, because the model's parameters are not convex or concave in $\mathbf{\Phi}_\text{p}$ in general, we have shown that the attacker's objective is also not convex or concave in general when $(\mathbf{\Phi}_\text{p}, \mathbf{y}_\text{p})$ are optimised simultaneously. Given this result, we hypothesise that most of the critical points of the attacker's objective are indeed saddle points, which in practice can slow down the optimisation process of the poisoning points, as mentioned in Ch. 3 and 5. We will explore this hypothesis in the next section.

6.3 Most of the Critical Points of the Attacker's Objective Function are Saddle Points

Understanding the nature of the critical points of the attacker's objective function, $\mathcal{A}(\mathbf{\Phi}_\text{p}, \mathbf{y}_\text{p})$, can foster further research to improve the performance of hypergradient-based algorithms, e.g., to speed up their convergence. For instance, in Ch. 3 and Ch. 5 the hypergradients for the poisoning points were normalised with respect to their ℓ_2-norm to accelerate their optimisation,. In this regard, we hypothesise that most of the critical points of the attacker's objective are indeed saddle points.

A critical point $(\mathbf{\Phi}_\text{p}^\star, \mathbf{y}_\text{p}^\star)$ of $\mathcal{A}(\mathbf{\Phi}_\text{p}, \mathbf{y}_\text{p})$ simultaneously satisfies the stationarity conditions in both $\mathbf{\Phi}_\text{p}$ and $\mathbf{y}_\text{p}$, i.e., the hypergradients in both sets of variables are zero. This means that, even if $\mathcal{A}(\mathbf{\Phi}_\text{p})$ has multiple critical points in $\mathbf{\Phi}_\text{p}$, i.e., when considering $\mathbf{y}_\text{p}$ fixed, they do not necessarily correspond to actual critical points of $\mathcal{A}(\mathbf{\Phi}_\text{p}, \mathbf{y}_\text{p})$.

Specifically, the stationarity conditions in $\mathbf{y}_\text{p}$—which correspond to the local minimum

[9] The function $\text{sgn}(\cdot)$ denotes the sign function.

of $\mathcal{A}(\mathbf{y}_\mathrm{p})$ when considering $\mathbf{\Phi}_\mathrm{p}$ fixed—are:

$$\mathbf{\Phi}^*_{\mathrm{p}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^\mathsf{T}_{\mathrm{val}_e}\left(\mathbf{y}_\mathrm{val}-\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{\Phi}^\star_\mathrm{p},\mathbf{y}^\star_\mathrm{p})\right)=\mathbf{0}_{n_\mathrm{p}}, \tag{6.17}$$

which can be derived from (6.20). Assuming $\mathcal{A}(\mathbf{\Phi}_\mathrm{p},\mathbf{y}_\mathrm{p})$ is strongly convex in $\mathbf{y}_\mathrm{p}$, we can write:

$$\begin{aligned}\mathbf{y}^\star_\mathrm{p}=&\left(\left(\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^{*\mathsf{T}}_{\mathrm{p}_e}\right)^\mathsf{T}\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^{*\mathsf{T}}_{\mathrm{p}_e}\right)^{-1}\\&\mathbf{\Phi}^\star_{\mathrm{p}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^\mathsf{T}_{\mathrm{val}_e}\left(\mathbf{y}_\mathrm{val}-\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^\mathsf{T}_{\mathrm{tr}_e}\mathbf{y}_\mathrm{tr}\right)\\=&\left(\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^{*\mathsf{T}}_{\mathrm{p}_e}\right)^\dagger\left(\mathbf{y}_\mathrm{val}-\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^\mathsf{T}_{\mathrm{tr}_e}\mathbf{y}_\mathrm{tr}\right),\end{aligned} \tag{6.18}$$

where $(\,\cdot\,)^\dagger$ denotes the Moore-Penrose pseudoinverse matrix.

If this point also satisfies the stationarity conditions $\forall\phi_{\mathrm{p}_{i,j}}\in\mathbf{\Phi}_\mathrm{p}$, where $i=1,\ldots,n_\mathrm{p}$ denotes the poisoning point and $j=1,\ldots,d-1$ denotes the basis function (see Eq. (6.9)):

$$\begin{aligned}&\left(\left(y^\star_{\mathrm{p}_i}-\boldsymbol{\phi}^{*\mathsf{T}}_{\mathrm{p}_{e_i}}\mathbf{w}^\star(\mathbf{\Phi}^\star_\mathrm{p},\mathbf{y}^\star_\mathrm{p})\right)\mathbf{e}_{j+1}-w^\star_j(\mathbf{\Phi}^\star_\mathrm{p},\mathbf{y}^\star_\mathrm{p})\boldsymbol{\phi}^\star_{\mathrm{p}_{e_i}}\right)^\mathsf{T}\mathbf{S}_\lambda(\mathbf{\Phi}^\star_\mathrm{p})^{-1}\mathbf{\Phi}^\mathsf{T}_{\mathrm{val}_e}\\&\left(\mathbf{y}_\mathrm{val}-\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{\Phi}^\star_\mathrm{p},\mathbf{y}^\star_\mathrm{p})\right)=0,\end{aligned} \tag{6.19}$$

the point can correspond to either local minima of $\mathcal{A}(\mathbf{\Phi}_\mathrm{p},\mathbf{y}_\mathrm{p})$, if it corresponds to local minima for all the entries of $\mathbf{\Phi}_\mathrm{p}$, or saddle points, if the point does not correspond to local minima for all the entries of $\mathbf{\Phi}_\mathrm{p}$. We hypothesise that, the larger d and n_p, the more likely it is that $\mathbf{y}^\star_\mathrm{p}$ corresponds to saddle points of $\mathcal{A}(\mathbf{\Phi}_\mathrm{p},\mathbf{y}_\mathrm{p})$.

To see this, recall that a necessary and sufficient condition for a critical point to be a local minimum is that the Hessian matrix evaluated at that point is positive semidefinite. On the contrary, a critical point is a local maximum if the Hessian matrix evaluated at that point is negative semidefinite. A symmetric matrix of size n_p is positive semidefinite—conversely, negative semidefinite—if all of its eigenvalues are non-negative—conversely, non-positive. Now, consider a random Hessian matrix. For instance, if the eigenvalues of the Hessian are generated independently according to a distribution symmetric around zero, a possible estimate of the probability of the Hessian being positive semidefinite or negative semidefinite is $(\frac{1}{2})^{n_\mathrm{p}d-1}$. However, when the size of the matrix is large, due to the correlations between eigenvalues and the laws governing large deviations of eigenvalues, specifically the smallest and largest eigenvalues (Fey et al., 2008), the probability that all of the eigenvalues are non-negative or all of the eigenvalues are non-positive is very low. Therefore, when n_p, d, or both are large, the probability of the critical point being a saddle

point is much larger than being a local minimum or a local maximum.

Although gradient ascent almost always escapes saddle points asymptotically (Lee et al., 2016), even with fairly natural random initialisation schemes and non-pathological functions, gradient ascent can be significantly slowed down by saddle points, taking exponential time to escape (Du et al., 2017). Three possible heuristics that can help alleviate this issue are: (i) gradient ascent with perturbations (Ge et al., 2015; Jin et al., 2017); (ii) gradient normalisation (Murray et al., 2019); (iii) combination of gradient normalisation with perturbations (Levy, 2016; Zhao et al., 2021). In this thesis (Ch. 3 and 5), we have followed the second approach. The intuition is as follows: a normalised gradient preserves the direction of the gradient but ignores magnitude, which leads to constant speed in the optimisation algorithm. Since the normalised gradient does not vanish near saddle points (Murray et al., 2019), the normalised gradient ascent algorithm should not slow down in the neighbourhood of saddle points and should therefore escape "quickly". In the experiments in Ch. 3 and 5, we have found this method to be able to skip most of the saddle points and speed up the optimisation process.

6.4 Asymptotic Behaviour of the Attacker's Objective Function

We shall discuss the asymptotic behaviour of the effectiveness objective function corresponding to the indiscriminate attack. Since this objective is a surrogate of the defender's test MSE, analysing its asymptotic behaviour can be useful to understand the impact of attack on the defender's model. Moreover, this analysis can also help understand the effect of the attack when the model is over-regularised. Effectively, this study supplements the empirical results of Ch. 5.

We recall that the expression used for the effectiveness objective is:

$$\mathcal{A}(\mathbf{\Phi}_{\mathrm{p}}, \mathbf{y}_{\mathrm{p}}) = \frac{1}{2n_{\mathrm{val}}} ||\mathbf{y}_{\mathrm{val}} - \mathbf{\Phi}_{\mathrm{val}_e} \mathbf{w}^{\star}(\mathbf{\Phi}_{\mathrm{p}}, \mathbf{y}_{\mathrm{p}})||_2^2. \tag{6.20}$$

We will analyse its asymptotic behaviour both in the labels and in the basis functions. Then, we will also investigate the impact of the attack when the target model is over-regularised.

6.4.1 Asymptotic Behaviour in the Poisoning Labels

Assuming $\mathbf{\Phi}_\mathrm{p}$ fixed, if $\mathcal{A}(\mathbf{y}_\mathrm{p})$ is strongly convex—equivalently, strictly convex for quadratic functions—in $\mathbf{y}_\mathrm{p}$, the limit when the poisoning labels become very large exists. In that case, we can express the poisoning labels as $\mathbf{y}_\mathrm{p} = \varepsilon_{y_\mathrm{p}} \mathbf{1}_{n_\mathrm{p}}$, where $\varepsilon_{y_\mathrm{p}} \in \mathbb{R}$ is an unbounded perturbation. Then:

$$\lim_{\varepsilon_{y_\mathrm{p}} \to \pm\infty} \mathcal{A}(\mathbf{y}_\mathrm{p} = \varepsilon_{y_\mathrm{p}} \mathbf{1}_{n_\mathrm{p}}) = \lim_{\varepsilon_{y_\mathrm{p}} \to \pm\infty} \frac{1}{2n_\mathrm{val}} \left|\left| \mathbf{y}_\mathrm{val} - \mathbf{\Phi}_{\mathrm{val}_e} \mathbf{S}_\lambda^{-1} \left(\mathbf{\Phi}_{\mathrm{tr}_e}^\intercal \mathbf{y}_\mathrm{tr} + \varepsilon_{y_\mathrm{p}} \mathbf{\Phi}_{\mathrm{p}_e}^\intercal \mathbf{1}_{n_\mathrm{p}} \right) \right|\right|_2^2 = \infty, \tag{6.21}$$

i.e., the attacker is unbounded and can achieve an arbitrary large error in the model, as expected.[10]

6.4.2 Asymptotic Behaviour in the Basis Functions

For the poisoning design matrix, because $\mathcal{A}(\mathbf{\Phi}_\mathrm{p})$ is not jointly convex in $\mathbf{\Phi}_\mathrm{p}$—or marginally convex in $\phi_{\mathrm{p}_{i,j}}$, as seen in Subsect. 6.2.2—there is not a unique way to define the limit when the entries of $\mathbf{\Phi}_\mathrm{p}$ become very large, and the limit may not exist. In order for this limit to exist, the objective function must be approaching the same value regardless of the path taken in $\mathbf{\Phi}_\mathrm{p}$ when moving towards the limit. For instance, when $\mathbf{\Phi}_{\mathrm{p}_e} = \mathbf{0}_{n_\mathrm{p} \times d}$ and $\mathbf{y}_\mathrm{p} = \mathbf{0}_{n_\mathrm{p}}$, $\mathcal{A}(\mathbf{\Phi}_\mathrm{p}, \mathbf{y}_\mathrm{p})$ approaches the validation MSE when the model is trained on clean data.

Further, note that the condition number of $\mathbf{S}_\lambda$ can change depending on the values of $\mathbf{\Phi}_\mathrm{p}$. For instance, a possible attacker's goal—which we will not explore in this thesis—can be to increase the condition number of $\mathbf{S}_\lambda$, which can worsen the performance of gradient-based algorithms at training time.[11]

Let us now assume a rank-1 poisoning attack, i.e., when all the poisoning points are linearly dependent in the basis functions space, which includes the cases where all the poisoning points are equal, or when there is only one poisoning point. In particular, let us assume that the poisoning design matrix has the following form: $\mathbf{\Phi}_\mathrm{p} = \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{n_\mathrm{p} \times (d-1)}$,

[10] If $\mathcal{A}(\mathbf{y}_\mathrm{p})$ is convex but not strongly convex, technically speaking, the limit may not exist. For example, for the function of two variables: $x, y, f(x, y) \in \mathbb{R}$: $f(x, y) = x^2$, $\nexists \lim_{y \to \infty} f(x, y)$.

[11] Generally speaking, in a rank-c_p attack there are $c_\mathrm{p} \leq d$ poisoning points linearly independent in the basis functions space, which can increase the condition number of the scatter matrix. For instance, in the particular case where there are only two different poisoning points (Koh et al., 2022), the attack corresponds to either a rank-1 attack—if the poisoning points are collinear—or a rank-2 attack—if they are non-collinear. Therefore, attacks of different rank are not generally equivalent.

where $\varepsilon_{\phi_\mathrm{p}} \in \mathbb{R}$ is an unbounded perturbation. Thus,

$$\begin{aligned}\boldsymbol{\Phi}_{\mathrm{p}_e}^\intercal \boldsymbol{\Phi}_{\mathrm{p}_e} &= \begin{bmatrix} \mathbf{1}_{n_\mathrm{p}}^\intercal \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{(d-1)\times n_\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{1}_{n_\mathrm{p}} & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{n_\mathrm{p}\times(d-1)} \end{bmatrix} = n_\mathrm{p} \begin{bmatrix} 1 & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1}^\intercal \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} & \varepsilon_{\phi_\mathrm{p}}^2 \mathbf{1}_{(d-1)\times(d-1)} \end{bmatrix} \\ &= n_\mathrm{p} \begin{bmatrix} 1 \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} \end{bmatrix} \begin{bmatrix} 1 & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1}^\intercal \end{bmatrix}.\end{aligned} \tag{6.22}$$

Recall that, using the Sherman-Morrison formula (Appx. A.2) under the assumption that $\mathbf{S}_{\mathrm{tr}_\lambda}$ is non-singular, the poisoned parameters can be written as:

$$\begin{aligned}\mathbf{w}^\star(\boldsymbol{\Phi}_\mathrm{p} = \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{n_\mathrm{p}\times(d-1)}) &= \left(\mathbf{S}_{\mathrm{tr}_\lambda} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\intercal \boldsymbol{\Phi}_{\mathrm{p}_e}\right)^{-1} \left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\intercal \mathbf{y}_\mathrm{tr} + \boldsymbol{\Phi}_{\mathrm{p}_e}^\intercal \mathbf{y}_\mathrm{p}\right) \\ &= \left(\mathbf{I}_d - \frac{n_\mathrm{p} \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \begin{bmatrix} 1 & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1}^\intercal \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} & \varepsilon_{\phi_\mathrm{p}}^2 \mathbf{1}_{(d-1)\times(d-1)} \end{bmatrix}}{1 + n_\mathrm{p} \begin{bmatrix} 1 & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1}^\intercal \end{bmatrix} \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \begin{bmatrix} 1 \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} \end{bmatrix}} \right) \\ &\quad \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^\intercal \mathbf{y}_\mathrm{tr} + \sum_{i=1}^{n_\mathrm{p}} y_{\mathrm{p}_i} \begin{bmatrix} 1 \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} \end{bmatrix} \right) \\ &= \frac{1}{1 + n_\mathrm{p} d_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2)} \left(\mathbf{I}_d + n_\mathrm{p} \left(d_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2) \mathbf{I}_d - \mathbf{D}_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2) \right) \right) \\ &\quad \left(\mathbf{w}_{\mathrm{tr}_\lambda}^\star + \sum_{i=1}^{n_\mathrm{p}} y_{\mathrm{p}_i} \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \begin{bmatrix} 1 \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} \end{bmatrix} \right)\end{aligned} \tag{6.23}$$

where $\mathbf{S}_{\mathrm{tr}_\lambda} = \mathbf{S}_\mathrm{tr} + \lambda \mathbf{I}_{d_e}$, $d_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2) = \begin{bmatrix} 1 & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1}^\intercal \end{bmatrix} \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \begin{bmatrix} 1 \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} \end{bmatrix}$, $\mathbf{D}_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2) = \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \begin{bmatrix} 1 & \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1}^\intercal \\ \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{d-1} & \varepsilon_{\phi_\mathrm{p}}^2 \mathbf{1}_{(d-1)\times(d-1)} \end{bmatrix}$, and $\mathbf{w}_{\mathrm{tr}_\lambda}^\star = \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} \boldsymbol{\Phi}_{\mathrm{tr}_e}^\intercal \mathbf{y}_\mathrm{tr}$ are the clean parameters. From (6.23), we can see that each element in $\mathbf{w}^\star$ is a rational function of $\varepsilon_{\phi_\mathrm{p}}$: $w_i^\star(\varepsilon_{\phi_\mathrm{p}}) = \frac{p_1(\varepsilon_{\phi_\mathrm{p}}^3)}{p_2(\varepsilon_{\phi_\mathrm{p}}^2)}$, $i = 0, \ldots, d-1$, where $p_k(\varepsilon_{\phi_\mathrm{p}}^q)$, $k = 1, 2$, denotes a polynomial in $\varepsilon_{\phi_\mathrm{p}}$ of degree q. There is one exception: the corresponding row of $\mathbf{D}_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2)$, $[\mathbf{D}_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2)]_{i,0:d-1} = d_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2) \mathbf{e}_{i+1}^\intercal$, in which case $w_i^\star(\varepsilon_{\phi_\mathrm{p}}) = \frac{p_1(\varepsilon_{\phi_\mathrm{p}})}{p_2(\varepsilon_{\phi_\mathrm{p}}^2)}$. Note that the latter case is only true when $d = 1$, because $\mathbf{D}_\mathrm{p}(\varepsilon_{\phi_\mathrm{p}}^2)$ reduces to a scalar.

Given the previous result and assumptions, we can now compute the limit when the

perturbation is very large:

$$\lim_{\varepsilon_{\phi_\mathrm{p}}\to\pm\infty}\mathcal{A}(\boldsymbol{\Phi}_\mathrm{p}=\varepsilon_{\phi_\mathrm{p}}\mathbf{1}_{n_\mathrm{p}\times(d-1)})=\lim_{\varepsilon_{\phi_\mathrm{p}}\to\pm\infty}\frac{1}{2n_\mathrm{val}}\left|\left|\mathbf{y}_\mathrm{val}-\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\boldsymbol{\Phi}_\mathrm{p}=\varepsilon_{\phi_\mathrm{p}}\mathbf{1}_{n_\mathrm{p}\times(d-1)})\right|\right|_2^2$$
$$=\begin{cases}\frac{1}{2n_\mathrm{val}}||\mathbf{y}_\mathrm{val}||_2^2 & \text{if } d=1,\\ \infty & \text{otherwise.}\end{cases} \tag{6.24}$$

The intuition of the result in (6.24) is: when $d=1$ and there is no bias term, $\mathbf{w}^\star=w^\star$ coincides with the slope of the regression line. In the Cartesian plane, if the poisoning points have a very large value in the x-axis, the slope will tend to zero to try to fit the poisoning points, which explains this particular case. If $d>1$, some or all the parameters will grow to $\pm\infty$ to try to fit the poisoning points, so the attacker is unbounded. Similar to the result obtained in Subsect. 6.4.1, this result is expected since the attacker can attain an arbitrary large error if the input is arbitrary large. We will finish this section by exploring the impact of the attack when the model is over-regularised.

6.4.3 Over-Regularised Ridge Regression under Data Poisoning

Here we shall theoretically analyse the effect of over-regularising the model, i.e., when the L_2 regularisation hyperparameter, $\lambda>0$, is very large. Intuitively, a very large value of λ can help mitigate the attack, as shown in Ch. 3 for classification models. This theoretical analysis also extends the empirical results of Ch. 5.

To understand the effect of L_2 regularisation, we can write the regularised scatter matrix, $\mathbf{S}_\lambda$, as:

$$\mathbf{S}_\lambda=\boldsymbol{\Phi}_e^\mathsf{T}\boldsymbol{\Phi}_e+\lambda\mathbf{I}_{d_e}=\begin{bmatrix}\mathbf{1}_n^\mathsf{T}\\ \boldsymbol{\Phi}^\mathsf{T}\end{bmatrix}\begin{bmatrix}\mathbf{1}_n & \boldsymbol{\Phi}\end{bmatrix}+\lambda\begin{bmatrix}0 & \mathbf{0}_{d-1}^\mathsf{T}\\ \mathbf{0}_{d-1} & \mathbf{I}_{(d-1)\times(d-1)}\end{bmatrix}$$
$$=\begin{bmatrix}n & \mathbf{1}_n^\mathsf{T}\boldsymbol{\Phi}\\ \boldsymbol{\Phi}^\mathsf{T}\mathbf{1}_n & \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Phi}+\lambda\mathbf{I}_{d-1}\end{bmatrix}, \tag{6.25}$$

where $n=n_\mathrm{tr}+n_\mathrm{p}$. Then, using the formula for the inverse of a block matrix (Appx. B),

and noticing that $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I}_{d-1}$ is always non-singular (Appx. D), we obtain:

$$\mathbf{S}_\lambda^{-1} = \begin{bmatrix} \left(n - \mathbf{1}_n^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I}_{d-1}\right)^{-1}\boldsymbol{\Phi}^\top\mathbf{1}_n\right)^{-1} & \mathbf{0}_{d-1}^\top \\ \mathbf{0}_{d-1} & \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I}_{d-1} - n^{-1}\boldsymbol{\Phi}^\top\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{\Phi}\right)^{-1} \end{bmatrix} \begin{bmatrix} 1 & -\mathbf{1}_n^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I}_{d-1}\right)^{-1} \\ -n^{-1}\boldsymbol{\Phi}^\top\mathbf{1}_n & \mathbf{I}_{d-1} \end{bmatrix}. \tag{6.26}$$

To compute the limits, we shall take into account that, for a matrix $\mathbf{A} \in \mathbb{R}^{(d-1)\times(d-1)}$:

$$\lim_{\lambda\to\infty}\left(\mathbf{A} + \lambda\mathbf{I}_{d-1}\right)^{-1} = \lim_{\lambda\to\infty}\lambda^{-1}\mathbf{I}_{d-1} = \mathbf{0}_{(d-1)\times(d-1)}. \tag{6.27}$$

Thus, when λ is very large, $\mathbf{S}_\lambda^{-1}$ becomes:

$$\begin{aligned} \lim_{\lambda\to\infty}\mathbf{S}_\lambda^{-1} &= \begin{bmatrix} n^{-1} & \mathbf{0}_{d-1}^\top \\ \mathbf{0}_{d-1} & \mathbf{0}_{(d-1)\times(d-1)} \end{bmatrix}\begin{bmatrix} 1 & \mathbf{0}_{d-1}^\top \\ -n^{-1}\boldsymbol{\Phi}_{\text{tr}}^\top\mathbf{1}_{\text{tr}} & \mathbf{I}_{d-1} \end{bmatrix} \\ &= \begin{bmatrix} n^{-1} & \mathbf{0}_{d-1}^\top \\ \mathbf{0}_{d-1} & \mathbf{0}_{(d-1)\times(d-1)} \end{bmatrix} = (n_{\text{tr}} + n_{\text{p}})^{-1}\mathbf{e}_1\mathbf{e}_1^\top. \end{aligned} \tag{6.28}$$

Next, we will make use of this result to compute the asymptotic behaviour of the attacker's objective function in both the poisoning labels and the poisoning basis functions.

Asymptotic Behaviour in the Poisoning Labels

Consequently, when combining a very large value of λ and very large perturbations in the poisoning labels, we obtain:

$$\begin{aligned} \lim_{\varepsilon_{y_{\text{p}}}\to\pm\infty,\lambda\to\infty}\mathcal{A}(\mathbf{y}_{\text{p}} = \varepsilon_{y_{\text{p}}}\mathbf{1}_{n_{\text{p}}}) = \lim_{\varepsilon_{y_{\text{p}}}\to\pm\infty}\frac{1}{2n_{\text{val}}}\Big|\Big|\mathbf{y}_{\text{val}} - (n_{\text{tr}} + n_{\text{p}})^{-1}\boldsymbol{\Phi}_{\text{val}_e}\mathbf{e}_1\mathbf{e}_1^\top \\ \left(\boldsymbol{\Phi}_{\text{tr}_e}^\top\mathbf{y}_{\text{tr}} + \varepsilon_{y_{\text{p}}}\boldsymbol{\Phi}_{\text{p}_e}^\top\mathbf{1}_{n_{\text{p}}}\right)\Big|\Big|_2^2 = \infty. \end{aligned} \tag{6.29}$$

This result suggests that, even if the model is over-regularised, the attacker is unbounded when perturbing the poisoning labels, because the bias term can grow to infinity. However, if we hypothetically set a different regularisation hyperparameter, $\lambda_0 > 0$, for the bias

term, the regularised scatter matrix $\mathbf{S}_{\lambda,\lambda_0}$ has the following form:

$$\mathbf{S}_{\lambda,\lambda_0} = \boldsymbol{\Phi}_e^\top \boldsymbol{\Phi}_e + \lambda \mathbf{I}_{d_e} + \lambda_0 \mathbf{e}_1 \mathbf{e}_1^\top = \begin{bmatrix} n + \lambda_0 & \mathbf{1}_n^\top \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^\top \mathbf{1}_n & \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I}_{d-1} \end{bmatrix}. \tag{6.30}$$

Again, if λ and λ_0 are very large, we obtain:

$$\lim_{\lambda \to \infty, \lambda_0 \to \infty} \mathbf{S}_{\lambda,\lambda_0}^{-1} = \begin{bmatrix} 0 & \mathbf{0}_{d-1}^\top \\ \mathbf{0}_{d-1} & \mathbf{0}_{(d-1)\times(d-1)} \end{bmatrix} = \mathbf{0}_{d \times d}. \tag{6.31}$$

Which leads to:

$$\nexists \lim_{\varepsilon_{y_\mathrm{p}} \to \pm\infty, \lambda \to \infty, \lambda_0 \to \infty} \mathcal{A}(\mathbf{y}_\mathrm{p} = \varepsilon_{y_\mathrm{p}} \mathbf{1}_{n_\mathrm{p}}). \tag{6.32}$$

The previous result implies that the attacker is not guaranteed to be unbounded when perturbing the poisoning labels, if the model is over-regularised and has a different regularisation hyperparameter for the bias term. In contrast to the usual settings in ML models, where regularising the bias term can introduce a significant amount of underfitting (Goodfellow et al., 2016), this suggests that using an optimal value of λ_0 for the bias term can help mitigate the effect of the attack. However, this assumption is out of the scope of this thesis, and its empirical evaluation is therefore left as future work.

Asymptotic Behaviour in the Basis Functions

When λ is very large, $\mathbf{S}_{\mathrm{tr}_\lambda}^{-1}$ becomes:

$$\lim_{\lambda \to \infty} \mathbf{S}_{\mathrm{tr}_\lambda}^{-1} = n_{\mathrm{tr}}^{-1} \mathbf{e}_1 \mathbf{e}_1^\top. \tag{6.33}$$

In consequence, when combining a very large value of λ and very large perturbations in the basis functions—see Eq. (6.23)—one comes to the conclusion that:

$$\nexists \lim_{\varepsilon_{\phi_\mathrm{p}} \to \pm\infty, d>1, \lambda \to \infty} \mathcal{A}(\boldsymbol{\Phi}_\mathrm{p} = \varepsilon_{\phi_\mathrm{p}} \mathbf{1}_{n_\mathrm{p} \times (d-1)}). \tag{6.34}$$

That the previous limits do not exist implies that the error of the poisoned model is not guaranteed to be unbounded even when the attacker's perturbations are unbounded. In a manner similar to the previous limit, this theoretically shows that regularisation can help alleviate the effect of poisoning attacks.

For simplicity, in the next sections of this chapter we shall assume the attacker only optimises the poisoning labels, i.e., that the attacker's objective is convex. We hypothesise that the approximation can also be good enough when considering the joint optimisation of poisoning labels and basis functions, if techniques such as gradient normalisation are applied. The underlying hypothesis is: because the optimisation process usually skips the saddle points—that arise due to the non-convexity in the basis functions and spread across the whole basis functions space—it focusses on the local maxima in the labels. Hence, there probably exist local maxima in the basis functions that coincides with the local maxima in the labels.

So far in this chapter, we have analysed the convexity, critical points, and asymptotic behaviour of the attacker's objective function. We have shown that this function is non-convex when considering simultaneously the poisoning labels and the poisoning basis functions. As a consequence, we have hypothesised that most of the critical points of the attacker's objectives are saddle points, which can slow down the optimisation process. For this reason, we normalised hypergradients w.r.t. the poisoning points in Ch. 3 and 5, to speed up the optimisation process of the poisoning points. Then, we have also investigated the asymptotic behaviour of the attacker's objective function, which supplements the empirical results obtained in Ch. 5, and also suggests that regularising the bias term of the model can help mitigate the attack.

Next, we shall show the equivalence between the optimal indiscriminate attack and CQP, to also provide some insights into the optimisation process and the solutions to the attacker's problem.

6.5 Equivalence Between the Optimal Indiscriminate Attack and Concave Quadratic Programming, If Only the Labels Are Optimised

If the attacker only modifies the labels of the poisoning points, and the attacker's objective function is convex, the optimal indiscriminate poisoning attack formulation without soft detectability constraints against ridge regression is mathematically equivalent to CQP (Chinchuluun et al., 2005). In CQP, the goal is to maximise a convex quadratic function or, equivalently, minimise a concave quadratic function, subject to the convex feasible

set. Even though the objective function is convex, CQP is, in general, non-convex and NP-Hard.

Recall that the optimal indiscriminate poisoning attack formulation against ridge regression is:

$$\begin{aligned} \max_{\mathbf{y}_\mathrm{p}} \quad & \mathcal{A}(\mathbf{y}_\mathrm{p}) \\ \text{s.t.} \quad & \mathbf{w}^\star(\mathbf{y}_\mathrm{p}) = \arg\min_{\mathbf{w}} \frac{1}{2(n_\mathrm{tr}+n_\mathrm{p})} \left(||\mathbf{y}_\mathrm{tr} - \mathbf{\Phi}_{\mathrm{tr}_e}\mathbf{w}||_2^2 + ||\mathbf{y}_\mathrm{p} - \mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}||_2^2 + \lambda||\mathbf{w}_{1:(d-1)}||_2^2 \right), \\ & \mathbf{y}_\mathrm{p} \in \mathcal{F}_{\mathcal{D}_\mathrm{p}}, \end{aligned} \tag{6.35}$$

where $\mathcal{A}(\mathbf{y}_\mathrm{p})$ is the same function as in (6.20)—considering $\mathbf{\Phi}_{\mathrm{p}_e}$ fixed—and $\mathcal{F}_{\mathcal{D}_\mathrm{p}}$ is the attacker's feasible set, assumed to be a closed bounded convex set. By replacing the inner optimisation problem with its KKT conditions, problem (6.35) can be rewritten as:

$$\begin{aligned} \max_{\mathbf{y}_\mathrm{p}} \quad & \frac{1}{n_\mathrm{val}} \left(\frac{1}{2}\mathbf{w}^\star(\mathbf{y}_\mathrm{p})^\mathsf{T}\mathbf{\Phi}_{\mathrm{val}_e}^\mathsf{T}\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) - \mathbf{y}_\mathrm{val}^\mathsf{T}\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) \right) \\ \text{s.t.} \quad & \mathbf{w}^\star(\mathbf{y}_\mathrm{p}) = \mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_\mathrm{p} + \mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr}, \\ & \mathbf{y}_\mathrm{p} \in \mathcal{F}_{\mathcal{D}_\mathrm{p}}, \end{aligned} \tag{6.36}$$

where $\mathbf{S}_\lambda = \mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{\Phi}_{\mathrm{tr}_e} + \lambda\mathbf{I}_{d_e} + \mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{\Phi}_{\mathrm{p}_e}$ is assumed to be positive definite. On the other hand, the general CQP formulation is:

$$\begin{aligned} \max_{\mathbf{y}_\mathrm{p}} \quad & \frac{1}{2}\mathbf{y}_\mathrm{p}^\mathsf{T}\mathbf{Q}\mathbf{y}_\mathrm{p} + \mathbf{c}^\mathsf{T}\mathbf{y}_\mathrm{p} \\ \text{s.t.} \quad & \mathbf{y}_\mathrm{p} \in \mathcal{F}_{\mathcal{D}_\mathrm{p}}, \end{aligned} \tag{6.37}$$

where $\mathbf{Q} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$ is a positive semidefinite matrix, $\mathbf{c} \in \mathbb{R}^{n_\mathrm{p}}$, and $\mathcal{F}_{\mathcal{D}_\mathrm{p}}$ is the feasible set of the problem, which is a closed bounded convex set. Thus, for the optimal poisoning attack formulation, $\mathbf{Q}$ and $\mathbf{c}$ correspond to:

$$\mathbf{Q} = \left(\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T} \right)^\mathsf{T} \mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}, \tag{6.38}$$

$$\mathbf{c} = \mathbf{\Phi}_{\mathrm{p}_e}\mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\mathrm{val}_e}^\mathsf{T} \left(\mathbf{\Phi}_{\mathrm{val}_e}\mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_\mathrm{tr} - \mathbf{y}_\mathrm{val} \right). \tag{6.39}$$

This equivalence can enable future research towards the use of CQP methods (Chinchuluun et al., 2005) to solve the attacker's problem , in contrast to hypergradient-based methods developed so far in the research community. Based on this equivalence, we shall

now explore some properties of the attacker's objective: number of combinations of the poisoning labels, and rank of the Hessian of the attacker's objective function. This analysis will allow us to further understand the optimisation process of the poisoning points.

6.5.1 On the Number of Combinations of Poisoning Labels

If the attacker's objective is convex, we can reason about the number of different combinations of the optimal poisoning labels in the attacker's search space. A well-known result in mathematical optimisation theory—and applicable to CQP—is that: if a function is convex, its maximum over a closed bounded (compact) convex set is achieved at an extreme point (boundary), i.e., a point in the set that is not a convex combination of any other points in the set. This intuitive result is known as Bauer's maximum principle (Bauer, 1958).

Therefore, in the context of optimal poisoning attacks, the optimal poisoning labels will always lie at the boundary of the attacker's feasible set if the attacker's objective function is convex and the feasible set is closed bounded and convex. This fact reduces the search space from the whole feasible set to its boundary.

On this matter, a typical choice for the closed bounded convex feasible set is a convex polytope. A convex polytope $\mathcal{P} \subseteq \mathbb{R}^{n_\mathrm{p}}$ is the set of all points that are convex combinations of finitely many points $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^{n_\mathrm{p}}$; we write $\mathcal{P} = \mathrm{Conv}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ and we refer to the points v_i as the vertices of $\mathcal{P}$. For instance, a convex polyhedron is a 3-dimensional convex polytope. Another example of convex polytope is a hyperrectangle or box, which is common in the data poisoning literature.

In particular, we shall show that, if the attacker's objective function $\mathcal{A}(\mathbf{y}_\mathrm{p})$ is convex in $\mathbf{y}_\mathrm{p}$, its maximum over the convex polytope $\mathcal{P}$ is achieved at one of its vertices.

Let $\mathbf{y}_{\mathrm{p}0}$ be a point in the attacker's feasible set. Then, because the feasible set is convex, $\mathbf{y}_{\mathrm{p}0}$ can be written as a convex combination of the vertices of $\mathcal{P}$, i.e., $\mathbf{y}_{\mathrm{p}0} = \sum_{i=1}^{k} \tau_i \mathbf{v}_i$ with $0 \leq \tau_i \leq 1$ and $\sum_{i=1}^{k} \tau_i = 1$. By Jensen's inequality:

$$\mathcal{A}(\mathbf{y}_{\mathrm{p}0}) = \mathcal{A}\left(\sum_{i=1}^{k} \tau_i \mathbf{v}_i\right) \leq \sum_{i=1}^{k} \tau_i \mathcal{A}(\mathbf{v}_i) \leq \max_i \mathcal{A}(\mathbf{v}_i). \tag{6.40}$$

In consequence, the maximum of $\mathcal{A}(\mathbf{y}_\mathrm{p})$ over $\mathbf{y}_\mathrm{p} \in \mathcal{P}$ is bounded above by the maximum over the vertices. Since the vertices belong to $\mathcal{P}$, the result holds true.

For example, if the feasible set is a box, the problem can have up to 2^{n_p} combinations of optimal poisoning labels, because a box has 2^{n_p} vertices and every vertex of the box can be a solution. In other words, the problem is combinatorially difficult.

Luckily, because $\mathcal{A}(\mathbf{y}_\mathrm{p})$ is differentiable in $\mathbf{y}_\mathrm{p}$, projected hypergradient ascent helps to alleviate this issue if only a local optimum is needed. In particular, if some eigenvalues of the hyper-Hessian are larger than others, the hypergradient will typically tend to align with the directions given by the eigenvectors corresponding to those top eigenvalues, which can drastically reduce the search space. For instance, if $n_\mathrm{p} = 2$, the number of vertices of the box constraint is four, but usually the optimisation algorithm will pick two of those vertices, which correspond to the vertices closest to the line defined by the top eigenvector and the local minimum of the objective function.

However, if the poisoning points are located in the vertices of the box, they can be easily detected by outlier detection techniques, as empirically shown in Ch. 5. This property justifies our novel stealth attack formulation. In particular, we will discuss in Sect. 6.6 more details on how the detectability constraints influences the optimal solutions.

Next, let us study another property of the CQP formulation, in terms of the Hessian of the attacker's objective.

6.5.2 Bounds on the Rank of the Hyper-Hessian Matrix and No. of Poisoning Points Optimised Simultaneously

In the experiments of Ch. 3 and 5, we used a fixed number of poisoning points simultaneously optimised—depending on the dataset—to be able to efficiently explore different ratios of poisoning points. In this respect, there is an aspect that has been somewhat overlooked in the literature, which relates to the optimal number of poisoning points optimised at the same time, where "optimal" here refers to the efficiency of the optimisation process. We will analyse this aspect—in the case where only the poisoning labels are optimised—from the standpoint of the hyper-Hessian (Hessian of the attacker's objective function).

A square matrix is singular if and only if one or more of its eigenvalues are zero. This issue translates into the optimisation domain as follows: if some eigenvalues of the hyper-Hessian are zero, the hypergradient is constant along the direction of their corresponding eigenvectors. In other words, if the objective function is convex, this condition implies that

the objective is not strictly convex.[12] In this case, the attacker's search space can have a lower dimensional structure. We formalise these intuitions in the following theorem.

Theorem 6.1. *Assume an optimal indiscriminate poisoning attack against ridge regression, where the attacker's objective is (6.20), and the attacker only modifies the labels of the poisoning points,* $\mathbf{y}_\mathrm{p}$. *Let the scatter matrix of the complete training data,* $\mathbf{S}_\lambda$, *be non-singular, let* $\mathbf{\Phi}_{\mathrm{val}_e}$ *have full-rank, and let and* $n_\mathrm{val} \geq d$, *where* d *is the number of basis functions (including the one corresponding to the bias term of the model). Then, the hyper-Hessian matrix of the attacker's objective,* $\mathbf{Q}$, *is non-singular if and only if* $\mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) = n_\mathrm{p}$.

Proof. A necessary and sufficient condition for $\mathbf{Q}$ to be non-singular is that $\mathrm{rank}(\mathbf{Q}) = n_\mathrm{p}$. Because $\mathbf{Q} = \frac{1}{n_\mathrm{val}} \left(\mathbf{\Phi}_{\mathrm{val}_e} \mathbf{S}_\lambda^{-1} \mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\right)^\mathsf{T} \mathbf{\Phi}_{\mathrm{val}_e} \mathbf{S}_\lambda^{-1} \mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}$, then $\mathrm{rank}(\mathbf{Q}) = \mathrm{rank}\left(\mathbf{\Phi}_{\mathrm{val}_e} \mathbf{S}_\lambda^{-1} \mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\right) = \mathrm{rank}\left(\mathbf{\Phi}_{\mathrm{val}_e} \mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\right) \leq \min(\mathrm{rank}(\mathbf{\Phi}_{\mathrm{val}_e}), \mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}))$ (see proofs in Appx. E.1, Appx. E.2, and Appx. E.3). Combining this result with Sylvester's rank inequality, we can bound $\mathrm{rank}(\mathbf{Q})$ as:

$$\mathrm{rank}(\mathbf{\Phi}_{\mathrm{val}_e}) + \mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) - d \leq \mathrm{rank}(\mathbf{Q}) \leq \min(\mathrm{rank}(\mathbf{\Phi}_{\mathrm{val}_e}), \mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e})). \tag{6.41}$$

Moreover, if we assume that $\mathbf{\Phi}_{\mathrm{val}_e}$ has full-rank[13] and $n_\mathrm{val} \geq d$,[14] the lower and upper bound become:

$$\mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) \leq \mathrm{rank}(\mathbf{Q}) \leq \min(d, \mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e})), \tag{6.42}$$

where, using the rank-nullity theorem: $\mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) = d - \mathrm{nullity}(\mathbf{\Phi}_{\mathrm{p}_e})$.[15] Moreover, assuming $n_\mathrm{p} \geq 1$:

$$1 \leq \mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) \leq \min(n_\mathrm{p}, d). \tag{6.43}$$

The inequalities in (6.42) becomes equalities if and only if $\mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) \leq d$, which is always true according to the inequality (6.43). Therefore:

$$\mathrm{rank}(\mathbf{Q}) = \mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) \leq \min(n_\mathrm{p}, d). \tag{6.44}$$

[12] For quadratic functions, strict convexity (or strict concavity) is easily characterised: a quadratic function is strictly convex (or strictly concave) if and only if its Hessian matrix is positive definite (negative definite, respectively) (Boyd and Vandenberghe, 2004).

[13] A matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ that has rank $\min(n, m)$ is said to have full-rank; otherwise, the matrix is rank-deficient.

[14] That is, the number of validation points linearly independent in the basis functions space is equal to or greater than the number of basis functions (including the one corresponding to the bias term of the model), which is common in practice.

[15] See Appx. E.1.

Thus, under the assumptions stated previously, $\mathbf{Q}$ is non-singular if and only if $\text{rank}(\mathbf{Q}) = \text{rank}(\mathbf{\Phi}_{\text{p}_e}) = n_{\text{p}}$. □

In words, the hyper-Hessian of the attacker's objective is non-singular if and only if all the poisoning points are linearly independent in the basis functions space. If $n_{\text{p}} \geq 1$, then $d \geq 2$, assuming the model has a bias term. For example, if $d = 2$ (one basis function corresponding to the features and another one corresponding to the bias term), and there is only one poisoning point, then the condition is satisfied and the hyper-Hessian is non-singular. However, if there are more than d poisoning points, the number of poisoning points linearly independent in the basis functions space is, at most, equal to d, and hence the hyper-Hessian is singular. From an optimisation viewpoint, the latter case can be interpreted such that there is redundant information in the problem.

In this regard, the condition number of the hyper-Hessian can be defined[16] as the ratio of the largest to the smallest eigenvalue of the hyper-Hessian (assuming the hyper-Hessian is positive definite) (Boyd and Vandenberghe, 2004). This essentially measures the difference between the flattest direction of movement and the most curved. Therefore, a singular hyper-Hessian implies, in the limit, that the condition number approaches infinity, so that the hyper-Hessian is ill-conditioned. Then, optimisation algorithms such as projected hypergradient ascent can make slow progress[17] towards the optimum, because their convergence rate depends on the condition number of the hyper-Hessian (Boyd and Vandenberghe, 2004). Further theoretical analysis when the poisoning basis functions are also optimised has not been considered in this thesis and is left as future work.

So far, in this chapter we have analysed different aspects of the attacker's objective function when it does not consider detectability constraints: convexity, critical points, asymptotic behaviour, and equivalence with CQP. We have also related the lessons learnt with some observations obtained in Ch. 3 and 5. In the next section, we will move on to analysing the properties and challenges of the attacker's objective function with detectability constraints, proposed in Ch. 5.

[16] Indeed, there exist several definitions of the condition number of a matrix. Broadly speaking, the condition number of a matrix (or more generally any function) measures how much the output values of the matrix can change for a small change in the input. This is used to measure how sensitive a matrix is to changes or errors in the input space, and how much error in the output space results from an error in the input space. Hence, a problem with a low condition number is said to be well-conditioned, while a problem with a high condition number is said to be ill-conditioned.

[17] Note that projected gradient methods behave in a manner similar to standard gradient methods in the interior of the feasible set.

6.6 Soft Detectability Constraints Do Not Preserve (in General) the Convexity of the Attacker's Objective

In Ch. 5, we developed a new attack formulation with soft detectability constraints as a multiobjective optimisation problem: one objective corresponding to the standard effectiveness function, and another objective corresponding to soft detectability constraints, which we called detectability-risk function. We then used scalarisation to transform the multiobjective into a single objective, by using a convex combination of the effectiveness and detectability objectives. One of the challenges of this formulation was that when combining objectives in this way the correlation (redundancy) between objective functions can affect the search space structure and algorithm performance. The objectives should be neither not correlated at all nor very highly correlated, otherwise it can be difficult to achieve a smooth transition for this trade-off. To reduce this redundancy, we redefined the detectability risk as the product of two loss functions that penalise points that are near the bounds of a certain confidence interval.

In this chapter, we will dive into this aspect, in terms of the convexity of the indiscriminate attack objective with soft detectability constraints. We have already seen that the poisoned model's parameters are not convex in $\mathbf{\Phi}_\mathrm{p}$ in general, so we shall focus on the analysis of the convexity in $\mathbf{y}_\mathrm{p}$. When only the labels of the poisoning points are optimised, this objective function is:

$$\mathcal{A}_d(\mathbf{y}_\mathrm{p}) = \alpha\mathcal{A}(\mathbf{y}_\mathrm{p}) - (1-\alpha)\mathcal{R}(\mathbf{y}_\mathrm{p}), \tag{6.45}$$

where $\alpha \in [0,1]$ and $\mathcal{R}(\mathbf{y}_\mathrm{p})$ is the detectability-risk function of the attacker, which constraints the values of the poisoning points.

If $\mathcal{A}(\mathbf{y}_\mathrm{p})$ and $\mathcal{R}(\mathbf{y}_\mathrm{p})$ are convex, $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is not necessarily convex, because it is a weighted sum of a convex function and a concave (negative convex) function. This can be seen as follows: the second-order condition for convexity requires that the hyper-Hessian matrix (Hessian of the attacker's objective) of $\mathcal{A}_d$, $\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}_d(\mathbf{y}_\mathrm{p}) = \alpha\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}(\mathbf{y}_\mathrm{p}) - (1-\alpha)\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{R}(\mathbf{y}_\mathrm{p})$, be positive semidefinite. This is equivalent to:

$$\alpha\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}(\mathbf{y}_\mathrm{p}) \succeq (1-\alpha)\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{R}(\mathbf{y}_\mathrm{p}), \tag{6.46}$$

where $\succeq$ denotes the Loewner order, which is a partial order defined by the convex cone of positive semidefinite matrices.[18] For two positive semidefinite symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$, we write $\mathbf{A} \succeq \mathbf{B}$ to denote that $\mathbf{A} - \mathbf{B}$ is positive semidefinite and symmetric. Thus, the convexity of $\mathcal{A}_d$ depends on $\mathbf{y}_\mathrm{p}$ and α.

Let us assume the detectability objective (i.e., the detectability-risk function) is non-negative. A practical detectability-risk function may aim to minimise the distance between the poisoning labels and some target labels:

$$\mathcal{R}(\mathbf{y}_\mathrm{p}) = \frac{1}{2n_\mathrm{p}} ||\mathbf{y}_\mathrm{p} - \mathbf{y}_\mathrm{target}(\mathbf{y}_\mathrm{p})||_2^2, \tag{6.47}$$

where $\mathbf{y}_\mathrm{target}(\mathbf{y}_\mathrm{p})$ represents the vector of target labels, assumed to be constrained. These labels can depend on $\mathbf{y}_\mathrm{p}$, e.g., through the poisoned parameters: $\mathbf{y}_\mathrm{target}(\mathbf{y}_\mathrm{p}) = \mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p})$; or not, e.g., through some target parameters, $\mathbf{y}_\mathrm{target} = \mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star_\mathrm{target}$, or through the initial values of the poisoning labels before their optimisation, $\mathbf{y}_{\mathrm{p}_0}$: $\mathbf{y}_\mathrm{target} = \mathbf{y}_{\mathrm{p}_0}$.

Another example of detectability-risk function is the machine teaching objective that minimises the distance between the poisoned and the clean parameters:

$$\mathcal{R}(\mathbf{y}_\mathrm{p}) = \frac{1}{2d} ||\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) - \mathbf{w}^\star_\mathrm{tr}||_2^2. \tag{6.48}$$

We shall analyse the convexity of these detectability-risk functions and their convex combination with the effectiveness objective function. As usual, we assume that the scatter matrix $\mathbf{S}_\lambda$ is positive definite.

6.6.1 Example 1: Constraining the Perturbations of the Poisoning Labels with a Poisoned Model as a Reference

Let us first analyse the detectability-risk function that minimises the error of the poisoning points on the poisoned mode (see Eq. (6.47)):

$$\mathcal{R}(\mathbf{y}_\mathrm{p}) = \frac{1}{2n_\mathrm{p}} ||\mathbf{y}_\mathrm{p} - \mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2 = \frac{1}{n_\mathrm{p}} \left(\frac{1}{2} ||\mathbf{y}_\mathrm{p}||_2^2 - \mathbf{y}_\mathrm{p}^\mathsf{T} \mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p}) + \frac{1}{2} ||\mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2 \right), \tag{6.49}$$

Clearly, $||\mathbf{y}_\mathrm{p}||_2^2$ is convex and $||\mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2$ is also convex in $\mathbf{y}_\mathrm{p}$, as shown in Sub-

[18] The positive semidefinite cone is a proper cone. The interior of the positive semidefinite cone consists of the positive definite matrices.

sect. 6.2.1.[19] Let us analyse the convexity of $\mathbf{y}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})$:

$$\mathbf{y}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}}) = \mathbf{y}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{tr}} + \mathbf{y}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\mathbf{y}_{\mathrm{p}}, \tag{6.50}$$

which is a quadratic function in $\mathbf{y}_{\mathrm{p}}$. Its hyper-Hessian matrix, $2\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} = \sqrt{2}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1/2}\left(\sqrt{2}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1/2}\right)^{\mathsf{T}}$, is positive definite, since $\mathbf{S}_{\lambda}^{-1}$ and $\mathbf{S}_{\lambda}^{-1/2}$ are symmetric and positive definite. Therefore, $\mathbf{y}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})$ is convex in $\mathbf{y}_{\mathrm{p}}$. In consequence, the hyper-Hessian of $\mathcal{R}(\mathbf{y}_{\mathrm{p}})$ is:

$$\begin{aligned}\nabla^2_{\mathbf{y}_{\mathrm{p}}}\mathcal{R}(\mathbf{y}_{\mathrm{p}}) &= \frac{1}{n_{\mathrm{p}}}\left(\mathbf{I}_{n_{\mathrm{p}}} - 2\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} + \left(\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\right)^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\right)\\ &= \frac{1}{n_{\mathrm{p}}}\left(\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} - \mathbf{I}_{n_{\mathrm{p}}}\right)^{\mathsf{T}}\left(\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} - \mathbf{I}_{n_{\mathrm{p}}}\right),\end{aligned} \tag{6.51}$$

since $\nabla^2_{\mathbf{y}_{\mathrm{p}}}\frac{1}{2}||\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})||_2^2 = \left(\boldsymbol{\Phi}_{\mathrm{p}_e}\frac{\partial\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})}{\partial\mathbf{y}_{\mathrm{p}}}\right)^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\frac{\partial\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})}{\partial\mathbf{y}_{\mathrm{p}}}$ is positive semidefinite and $\frac{\partial\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})}{\partial\mathbf{y}_{\mathrm{p}}} = \mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}$. Thus, $\nabla^2_{\mathbf{y}_{\mathrm{p}}}\mathcal{R}(\mathbf{y}_{\mathrm{p}})$ is positive semidefinite and $\mathcal{R}(\mathbf{y}_{\mathrm{p}})$ is convex. $\mathcal{R}(\mathbf{y}_{\mathrm{p}})$ is strongly convex if and only if: $\mathrm{rank}\left(\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} - \mathbf{I}_{n_{\mathrm{p}}}\right) = n_{\mathrm{p}}$.

To determine the convexity of Eq. (6.45), we also need to calculate $\nabla^2_{\mathbf{y}_{\mathrm{p}}}\mathcal{A}(\mathbf{y}_{\mathrm{p}})$. Recall that $\mathcal{A}(\mathbf{y}_{\mathrm{p}}) = \frac{1}{2n_{\mathrm{val}}}||\mathbf{y}_{\mathrm{val}} - \boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})||_2^2 = \frac{1}{2n_{\mathrm{val}}}||\mathbf{y}_{\mathrm{val}}||_2^2 - \frac{1}{n_{\mathrm{val}}}\mathbf{y}_{\mathrm{val}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}}) + \frac{1}{2n_{\mathrm{val}}}||\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})||_2^2$. Then:

$$\nabla^2_{\mathbf{y}_{\mathrm{p}}}\mathcal{A}(\mathbf{y}_{\mathrm{p}}) = \frac{1}{n_{\mathrm{val}}}\nabla^2_{\mathbf{y}_{\mathrm{p}}}\frac{1}{2}||\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})||_2^2 = \frac{1}{n_{\mathrm{val}}}\left(\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\right)^{\mathsf{T}}\left(\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\right), \tag{6.52}$$

which is positive semidefinite and therefore $\mathcal{A}(\mathbf{y}_{\mathrm{p}})$ is convex in $\mathbf{y}_{\mathrm{p}}$, as shown also in Subsect. 6.2.1. Thus, from Eq. (6.46), we can conclude that $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is convex in $\mathbf{y}_{\mathrm{p}}$ if and only if:

$$\alpha\frac{n_{\mathrm{p}}}{n_{\mathrm{val}}}\left(\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\right)^{\mathsf{T}}\left(\boldsymbol{\Phi}_{\mathrm{val}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}}\right) \succeq (1-\alpha)\left(\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} - \mathbf{I}_{n_{\mathrm{p}}}\right)^{\mathsf{T}}\left(\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\lambda}^{-1}\boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} - \mathbf{I}_{n_{\mathrm{p}}}\right), \tag{6.53}$$

and concave otherwise. In words, the constraint on the values poisoning design matrix in Eq. (6.53) determines the convexity or concavity of the attacker's objective function.

[19] Note that $\mathbf{y}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})$ and $||\boldsymbol{\Phi}_{\mathrm{p}_e}\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})||_2^2$ are not convex or concave in $\boldsymbol{\Phi}_{\mathrm{p}}$ in general.

6.6.2 Example 2: Constraining the Perturbations of the Poisoning Labels with a Clean Model as a Reference

From Eq. (6.47), if $\mathbf{y}_{\text{target}}$ does not depend on $\mathbf{y}_{\text{p}}$, we obtain:

$$\mathcal{R}(\mathbf{y}_{\text{p}}) = \frac{1}{2n_{\text{p}}}||\mathbf{y}_{\text{p}} - \mathbf{y}_{\text{p}_0}||_2^2 = \frac{1}{n_{\text{p}}}\left(\frac{1}{2}||\mathbf{y}_{\text{p}}||_2^2 - \mathbf{y}_{\text{p}}^\mathsf{T}\mathbf{y}_{\text{p}_0} + \frac{1}{2}||\mathbf{y}_{\text{p}_0}||_2^2\right). \tag{6.54}$$

The hyper-Hessian of $\mathcal{R}(\mathbf{y}_{\text{p}})$ is:

$$\nabla_{\mathbf{y}_{\text{p}}}^2 \mathcal{R}(\mathbf{y}_{\text{p}}) = \frac{1}{n_{\text{p}}}\mathbf{I}_{n_{\text{p}}}, \tag{6.55}$$

which is positive definite. Therefore, $\mathcal{R}(\mathbf{y}_{\text{p}})$ is strongly convex, and $\mathcal{A}_d(\mathbf{y}_{\text{p}})$ is convex if:

$$\alpha\frac{n_{\text{p}}}{n_{\text{val}}}\left(\mathbf{\Phi}_{\text{val}_e}\mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\text{p}_e}^\mathsf{T}\right)^\mathsf{T}\mathbf{\Phi}_{\text{val}_e}\mathbf{S}_\lambda^{-1}\mathbf{\Phi}_{\text{p}_e}^\mathsf{T} \succeq (1-\alpha)\mathbf{I}_{n_{\text{p}}}, \tag{6.56}$$

and concave otherwise.

6.6.3 Example 3: Constraining the Perturbations of the Model's Parameters: Machine Teaching

As for the detectability objective in the form of machine teaching:

$$\mathcal{R}(\mathbf{y}_{\text{p}}) = \frac{1}{2d}||\mathbf{w}^\star(\mathbf{y}_{\text{p}}) - \mathbf{w}_{\text{tr}}^\star||_2^2 = \frac{1}{d}\left(\frac{1}{2}||\mathbf{w}^\star(\mathbf{y}_{\text{p}})||_2^2 - \mathbf{w}^\star(\mathbf{y}_{\text{p}})^\mathsf{T}\mathbf{w}_{\text{tr}}^\star + \frac{1}{2}||\mathbf{w}_{\text{tr}}^\star||_2^2\right), \tag{6.57}$$

its hyper-Hessian is:

$$\nabla_{\mathbf{y}_{\text{p}}}^2 \mathcal{R}(\mathbf{y}_{\text{p}}) = \frac{1}{d}\left(\frac{\partial \mathbf{w}^\star}{\partial \mathbf{y}_{\text{p}}}\right)^\mathsf{T}\frac{\partial \mathbf{w}^\star}{\partial \mathbf{y}_{\text{p}}} = \frac{1}{d}\mathbf{\Phi}_{\text{p}_e}\mathbf{S}_\lambda^{-2}\mathbf{\Phi}_{\text{p}_e}^\mathsf{T}, \tag{6.58}$$

which is positive semidefinite. It is positive definite if and only if $\text{rank}\left(\mathbf{\Phi}_{\text{p}_e}\mathbf{S}_\lambda^{-2}\mathbf{\Phi}_{\text{p}_e}^\mathsf{T}\right) = \text{rank}(\mathbf{\Phi}_{\text{p}_e}) = n_{\text{p}}$. Hence, $\mathcal{A}_d(\mathbf{y}_{\text{p}})$ is convex if and only if:

$$\alpha\frac{d}{n_{\text{val}}}\mathbf{\Phi}_{\text{val}_e}^\mathsf{T}\mathbf{\Phi}_{\text{val}_e} \succeq (1-\alpha)\mathbf{I}_d, \tag{6.59}$$

and concave otherwise. This shows that, in this case, the condition for convexity does not depend on the poisoning points but only on the validation set, d, and α.

6.6.4 Upper Bound on the Rank of the Hyper-Hessian when Considering Detectability Constraints

We can also calculate an upper bound on the rank of the hyper-Hessian when considering detectability constraints. If $\text{rank}(\nabla^2_{\mathbf{y}_\text{p}}\mathcal{A}_d(\mathbf{y}_\text{p})) = n_\text{p}$, then the hyper-Hessian is non-singular and therefore $\mathcal{A}_d$ is strictly convex or strictly concave, depending on the conditions obtained previously. Moreover, as we have seen before, because both hyper-Hessians $\nabla^2_{\mathbf{y}_\text{p}}\mathcal{A}(\mathbf{y}_\text{p})$ and $\nabla^2_{\mathbf{y}_\text{p}}\mathcal{R}(\mathbf{y}_\text{p})$ are positive semidefinite, they can be factorised as: $\nabla^2_{\mathbf{y}_\text{p}}\mathcal{A}(\mathbf{y}_\text{p}) = \mathbf{L}_\mathcal{A}\mathbf{L}_\mathcal{A}^\intercal$ and $\nabla^2_{\mathbf{y}_\text{p}}\mathcal{R}(\mathbf{y}_\text{p}) = \mathbf{L}_\mathcal{R}\mathbf{L}_\mathcal{R}^\intercal$, where $\mathbf{L}_\mathcal{A} \in \mathbb{R}^{n_\text{p}\times n_\text{val}}$, $\mathbf{L}_\mathcal{R} \in \mathbb{R}^{n_\text{p}\times k_\mathcal{R}}$, and $k_\mathcal{R}$ depends on the definition of $\mathcal{R}(\mathbf{y}_\text{p})$. For example, for the machine teaching objective (6.57), $k_\mathcal{R} = d$. Under the same assumptions as in Subsect. 6.5.2, i.e., $\mathbf{\Phi}_{\text{val}_e}$ has full-rank and $n_\text{val} \geq d$, and considering the property that the rank of the sum of matrices is subadditive (see Appx. E.4), the upper bound on the rank of the hyper-Hessian is:

$$\begin{aligned}\text{rank}\left(\nabla^2_{\mathbf{y}_\text{p}}\mathcal{A}_d(\mathbf{y}_\text{p})\right) &= \text{rank}\left(\alpha\nabla^2_{\mathbf{y}_\text{p}}\mathcal{A}(\mathbf{y}_\text{p}) - (1-\alpha)\nabla^2_{\mathbf{y}_\text{p}}\mathcal{R}(\mathbf{y}_\text{p})\right)\\ &\leq \text{rank}\left(\nabla^2_{\mathbf{y}_\text{p}}\mathcal{A}(\mathbf{y}_\text{p})\right) + \text{rank}\left(\nabla^2_{\mathbf{y}_\text{p}}\mathcal{R}(\mathbf{y}_\text{p})\right)\\ &= \text{rank}(\mathbf{\Phi}_{\text{p}_e}) + \text{rank}(\mathbf{L}_\mathcal{R}) \leq \min(n_\text{p}, d) + \min(n_\text{p}, k_\mathcal{R}),\end{aligned} \tag{6.60}$$

where it is manifest that the poisoning design matrix, $\mathbf{\Phi}_\text{p}$, plays again a crucial role in the rank of the hyper-Hessian. We can deduce that if $\mathcal{A}(\mathbf{y}_\text{p})$, $\mathcal{R}(\mathbf{y}_\text{p})$, or both are strictly convex, it does not guarantee that $\mathcal{A}_d(\mathbf{y}_\text{p})$ is strictly convex or strictly concave (or only convex or concave). However, if $d + k_\mathcal{R} < n_\text{p}$, which implies that both $\mathcal{A}(\mathbf{y}_\text{p})$ and $\mathcal{R}(\mathbf{y}_\text{p})$ are not strictly convex, then it is guaranteed that $\mathcal{A}_d(\mathbf{y}_\text{p})$ is not strictly convex or strictly concave. Considering the detectability-risk functions defined previously, it is only guaranteed that $\mathcal{A}_d(\mathbf{y}_\text{p})$ will not be strictly convex or strictly concave when using the machine teaching objective (6.57), if $2d < n_\text{p}$. That is, if the number of poisoning points is greater than two times the number of basis functions.

The consequence of the attacker's objective being strictly concave, apart from helping in convergence in hypergradient-based optimisation algorithms, is that it guarantees the uniqueness of optimal solutions. However, from the attacker's perspective, the uniqueness of solutions may not be desirable, as we shall see in Subsect. 6.7.2: if all the poisoning points are concentrated in the same region or even they are the same, they can be more easily detected.

6.6.5 Eigenvalues of the Hyper-Hessian Matrix and Convexity

In the previous examples, we analysed the convexity of the attacker's objective depending on the values of the poisoning points. Here, we also provide a complementary perspective and investigate its convexity in terms of the hyperparameter that controls the trade-off between effectiveness and detectability, α. In particular, we shall determine the value of α that triggers the transition from convexity to concavity. To do so, we will consider the eigenvalues of the effectiveness and detectability-risk functions.

We start from the definition of the hyper-Hessian being negative semidefinite—which implies that $\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is concave:

$$\mathbf{y}_\mathrm{p}^\mathsf{T} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}_d(\mathbf{y}_\mathrm{p}) \right) \mathbf{y}_\mathrm{p} = \mathbf{y}_\mathrm{p}^\mathsf{T} \left(\alpha \nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) - (1-\alpha) \nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}) \right) \mathbf{y}_\mathrm{p} \leq 0. \tag{6.61}$$

As in Appx. C, we can bound each term of the previous expression as:

$$\alpha \lambda_{\min} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right) ||\mathbf{y}_\mathrm{p}||_2^2 \leq \alpha \mathbf{y}_\mathrm{p}^\mathsf{T} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right) \mathbf{y}_\mathrm{p} \leq \alpha \lambda_{\max} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right) ||\mathbf{y}_\mathrm{p}||_2^2, \tag{6.62}$$

$$\begin{aligned} -(1-\alpha) \lambda_{\max} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right) ||\mathbf{y}_\mathrm{p}||_2^2 &\leq -(1-\alpha) \mathbf{y}_\mathrm{p}^\mathsf{T} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}) \right) \mathbf{y}_\mathrm{p} \\ &\leq -(1-\alpha) \lambda_{\min} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}) \right) ||\mathbf{y}_\mathrm{p}||_2^2, \end{aligned} \tag{6.63}$$

where $\lambda_{\min}(\,\cdot\,)$ and $\lambda_{\max}(\,\cdot\,)$ denote the smallest and largest eigenvalues of the corresponding hyper-Hessians. Therefore, the condition for concavity is:

$$\alpha \lambda_{\max} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right) - (1-\alpha) \lambda_{\min} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}) \right) \leq 0, \tag{6.64}$$

which leads to the following condition for α:

$$\alpha \leq (1 + \kappa_{\mathcal{A}/\mathcal{R}})^{-1}, \tag{6.65}$$

where we define the attacker's multiobjective condition number, $\kappa_{\mathcal{A}/\mathcal{R}}$, as:

$$\kappa_{\mathcal{A}/\mathcal{R}} = \frac{\lambda_{\max} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right)}{\lambda_{\min} \left(\nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}) \right)} = \frac{\left|\left| \nabla^2_{\mathbf{y}_\mathrm{p}} \mathcal{A}(\mathbf{y}_\mathrm{p}) \right|\right|_2}{\left|\left| \nabla^{-2}_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}) \right|\right|_2} \tag{6.66}$$

where $\nabla^{-2}_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p})$ is the inverse of the hyper-Hessian of the detectability-risk function, and $||\,\cdot\,||_2$ denotes here the spectral norm the corresponding matrix. It is noteworthy that

$\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is always convex when $\kappa_{\mathcal{A}/\mathcal{R}} \to \infty$ because, in this case, the condition for concavity leads to $\alpha \leq 0$, which is out or range. This happens when $\lambda_{\max}\left(\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}(\mathbf{y}_\mathrm{p})\right) \to \infty$ or $\lambda_{\min}\left(\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{R}(\mathbf{y}_\mathrm{p})\right) \to 0$, i.e., when either the spectral norm of $\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}(\mathbf{y}_\mathrm{p})$ is very large, or $\mathcal{R}(\mathbf{y}_\mathrm{p})$ is convex but not strongly convex. On the other hand, $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is always concave when $\kappa_{\mathcal{A}/\mathcal{R}} \to 0$, which leads to $\alpha \leq 1$, which is always true. This happens when $\lambda_{\max}\left(\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}(\mathbf{y}_\mathrm{p})\right) \to 0$ or $\lambda_{\min}\left(\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{R}(\mathbf{y}_\mathrm{p})\right) \to \infty$, i.e., when either the spectral norm of $\nabla^2_{\mathbf{y}_\mathrm{p}}\mathcal{A}(\mathbf{y}_\mathrm{p})$ is very low or the smallest eigenvalue of $\mathcal{R}(\mathbf{y}_\mathrm{p})$ is very large. In practice, a very large value of $\kappa_{\mathcal{A}/\mathcal{R}}$ leads to an objective function predominantly convex—the poisoning labels lie at the boundary of the feasible set. In contrast, a very low value of $\kappa_{\mathcal{A}/\mathcal{R}}$ means that the objective is predominantly concave—the poisoning labels lie in the local maximum of $-\mathcal{R}(\mathbf{y}_\mathrm{p})$, if the maximum is in the interior of the feasible set, or at its boundary otherwise.

In summary, we have seen that the convexity of the attacker's objective function can be brittle if the detectability-risk function is not carefully designed. We have provided two complementary perspectives on the convexity/concavity of the attacker's objective: (i) as a function of the values of the hyper-Hessian and the hyperparameter α (Subsect. 6.6.1, 6.6.2, and 6.6.3); and (ii) depending on an upper bound on the value of α (Subsect. 6.6.5). These lessons learnt will allow us to synthesise the main challenges of the optimal attack formulation with soft detectability constraints in the next section.

6.7 Challenges of the Optimal Attack Formulation with Soft Detectability Constraints

There are three main challenges in the optimal attack formulation with detectability constraints proposed in Ch. 5, which are related to second-order analysis (hyper-Hessian matrix and curvature), first-order analysis (hypergradient and diversity of poisoning points), and zeroth-order analysis (order of magnitude of the effectiveness and detectability-risk functions and trade-off between them). We shall discuss these three challenges and justify the detectability-risk function proposed in Ch. 5.

6.7.1 Second-Order Challenge: Curvature

The second-order challenge is related to the convexity of the attacker's objective: in all the examples analysed in Sect. 6.6, because the attacker maximises $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$, there exists an abrupt transition between constrained and unconstrained poisoning points: if $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is convex, the poisoning points lie at the extreme (boundary) of the attacker's feasible set; if $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is concave, the optimal poisoning labels, $\mathbf{y}_\mathrm{p}^\star$, lie in the local maximum of $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$. This phenomenon may not be desirable for attackers who want to achieve some diversity in the distribution of poisoning points. Here, diversity is related to the number of local optima of the attacker's objective.

6.7.2 First-Order Challenge: Attacker's KKT Conditions and Diversity of Poisoning Points

The first-order challenge is related to the idea that, if $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ is convex in $\mathbf{y}_\mathrm{p}$ (the optimal poisoning points lie at the boundary of the attacker's feasible set), or concave and quadratic in $\mathbf{y}_\mathrm{p}$, the diversity and therefore stealth of the optimal poisoning points can also be limited. We shall discuss why.

Let us assume that the inner minimisation problem (learning problem) of the attacker's problem is convex in the model's parameters and exhibits a unique solution from its stationarity conditions, as is the case of ridge regression. Let us also assume that the attacker's objective function is twice differentiable in $\mathbf{y}_\mathrm{p}$, and that the attacker's feasible set corresponds to a convex set defined by m_p inequality constraint functions in $\mathbf{y}_\mathrm{p}$: $\mathbf{g}_\mathrm{p}(\mathbf{y}_\mathrm{p}) = [g_{\mathrm{p}_1}(\mathbf{y}_\mathrm{p}) \ \dots \ g_{\mathrm{p}_{m_\mathrm{p}}}(\mathbf{y}_\mathrm{p})]^\mathsf{T} : g_{\mathrm{p}_i}(\mathbf{y}_\mathrm{p}) \leq 0, \ i = 1, \dots, m_\mathrm{p}$, which are differentiable *almost everywhere*, as usual in the data poisoning literature. Note that the attacker's feasible set is bounded, so a global maximum must exist: a continuous function on a closed and bounded set has a maximum there.

For instance, the ℓ_2-norm constraint $\frac{1}{2}||\mathbf{y}_\mathrm{p} - \mathbf{y}_{\mathrm{p}_0}||_2^2 \leq \varepsilon_2^2$, where $\varepsilon_2^2 \in \mathbb{R}_{>0}$, corresponds to $g_\mathrm{p}(\mathbf{y}_\mathrm{p}) = \frac{1}{2}||\mathbf{y}_\mathrm{p} - \mathbf{y}_{\mathrm{p}_0}||_2^2 - \varepsilon_2^2$. As another example, the box constraint $\mathbf{y}_{\mathrm{p}_\mathrm{min}} \preceq \mathbf{y}_\mathrm{p} \preceq \mathbf{y}_{\mathrm{p}_\mathrm{max}}$, where $\mathbf{y}_{\mathrm{p}_\mathrm{min}}, \mathbf{y}_{\mathrm{p}_\mathrm{max}} \in \mathbb{R}^{n_\mathrm{p}}$, corresponds to $\mathbf{g}_\mathrm{p}(\mathbf{y}_\mathrm{p}) = |\mathbf{y}_\mathrm{p} - \frac{1}{2}(\mathbf{y}_{\mathrm{p}_\mathrm{max}} + \mathbf{y}_{\mathrm{p}_\mathrm{min}})| - \frac{1}{2}(\mathbf{y}_{\mathrm{p}_\mathrm{max}} - \mathbf{y}_{\mathrm{p}_\mathrm{min}})$, where the absolute value function is applied element-wise. Note that, if $\frac{1}{2}(\mathbf{y}_{\mathrm{p}_\mathrm{max}} - \mathbf{y}_{\mathrm{p}_\mathrm{min}}) = \varepsilon_\infty \mathbf{1}_{n_\mathrm{p}}$, where $\varepsilon_\infty \in \mathbb{R}_{>0}$, then the box constraint corresponds to the ℓ_∞-norm: $g_\mathrm{p}(\mathbf{y}_\mathrm{p}) = ||\mathbf{y}_\mathrm{p} - \frac{1}{2}(\mathbf{y}_{\mathrm{p}_\mathrm{max}} + \mathbf{y}_{\mathrm{p}_\mathrm{min}})||_\infty - \varepsilon_\infty$.

Then, the optimality of the poisoning labels can be analysed from the KKT conditions of the attacker's problem (outer maximisation problem), which we shall describe next.

Attacker's KKT Conditions

Bear in mind that we do not aim to solve the attacker's problem by leveraging its KKT conditions. Instead, we aim to obtain some insights about the nature of the set of optimal poisoning labels. Under some regularity conditions, the KKT conditions are necessary conditions for the optimality of the Lagrangian of the attacker's objective function, $\mathcal{A}_{d_{\mathcal{L}}}(\mathbf{y}_{\mathrm{p}}, \boldsymbol{\mu}_{\mathrm{p}})$:

$$\mathcal{A}_{d_{\mathcal{L}}}(\mathbf{y}_{\mathrm{p}}, \boldsymbol{\mu}_{\mathrm{p}}) = \mathcal{A}_d(\mathbf{y}_{\mathrm{p}}) - \boldsymbol{\mu}_{\mathrm{p}}^{\mathsf{T}} \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}), \tag{6.67}$$

where $\boldsymbol{\mu}_{\mathrm{p}} = [\mu_{1_{\mathrm{p}}} \ \ldots \ \mu_{\mathrm{p}_{m_{\mathrm{p}}}}]^{\mathsf{T}}$ are the KKT multipliers associated with the attacker's inequality constraints. Then, the stationarity conditions of the Lagrangian of the attacker's objective are:

$$\mathbf{0}_{n_{\mathrm{p}}} \in \partial_{\mathbf{y}_{\mathrm{p}}} \left(\mathcal{A}_d(\mathbf{y}_{\mathrm{p}}^{\star}) - \boldsymbol{\mu}_{\mathrm{p}}^{\star\mathsf{T}} \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) \right), \tag{6.68}$$

where $\partial_{\mathbf{y}_{\mathrm{p}}}(\ \cdot \ )$ denotes the subdifferential.[20] Since the subdifferential is a set, Eq (6.68) means that the vector of zeros belongs to the subdifferential (Hastie et al., 2015). In the general case, the subdifferential of a sum of functions is not equal to the sum of the subdifferentials of the functions, because the subdifferential is not unique in general. However, under mild conditions on concave non-differentiable objective functions (in the case of maximisation problems) and convex non-differentiable constraint functions, and considering that $\mathrm{dom}(\mathcal{A}_d(\mathbf{y}_{\mathrm{p}}^{\star})) \cap \mathrm{dom}(-\boldsymbol{\mu}_{\mathrm{p}}^{\star\mathsf{T}} \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star})) \neq \varnothing$, Moreau-Rockafellar's theorem (Moreau, 1965; Rockafellar, 1966) allows to apply the KKT conditions using the sum of the subdifferentials.[21] On the other hand, if the objective function is non-concave but differentiable, and the constraints functions are convex and differentiable, the KKT conditions can also be applied, although in this case the KKT conditions are generally not sufficient for optimality of the solution (Boyd and Vandenberghe, 2004).[22] Hence,

[20] Given a convex function $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, the subdifferential is the set of all subgradients of g at the point $\boldsymbol{\beta}$. In geometric terms, the subgradient vector is the normal to a (non-vertical) hyperplane that supports the epigraph of g, denoted by $\partial_{\mathbf{x}} g(\boldsymbol{\beta})$. Whenever g is differentiable at $\boldsymbol{\beta}$, then the subdifferential reduces to a single vector—namely $\partial_{\mathbf{x}} g(\boldsymbol{\beta}) = \{\nabla_{\mathbf{x}} g(\boldsymbol{\beta})\}$. At points of non-differentiability, the subdifferential is a convex set containing all possible subgradients (Hastie et al., 2015).

[21] Generalised subdifferentials for non-concave objective functions (Rockafellar, 1980), out of the scope of this thesis, can also lead to alternative formulations.

[22] For instance, combining KKT conditions and second-order conditions can be sufficient (Boyd and Vandenberghe, 2004).

Eq. (6.68) reduces to:

$$\mathbf{0}_{n_\mathrm{p}} \in \nabla_{\mathbf{y}_\mathrm{p}} \mathcal{A}_d(\mathbf{y}_\mathrm{p}^\star) - \left(\partial_{\mathbf{y}_\mathrm{p}} \mathbf{g}_\mathrm{p}(\mathbf{y}_\mathrm{p}^\star)\right)^\mathsf{T} \boldsymbol{\mu}_\mathrm{p}^\star. \tag{6.69}$$

For simplicity, we assume that the subdifferentials are defined is such a way that they are single-valued at their discontinuities, instead of including all possible subderivatives. Moreover, we assume that the single value at each discontinuity is a convex combination of the one-sided derivatives at the discontinuity. For instance, for the sign function—which is the subdifferential of the absolute value function—we can define:

$$\operatorname{sgn}(x) = \partial_x |x| = \begin{cases} 1, & \text{if } x > 0, \\ 2\beta - 1, & \text{if } x = 0, \\ -1 & \text{if } x < 0, \end{cases} \tag{6.70}$$

where $\beta \in [0, 1]$. Under the previous assumption, or if $\mathbf{g}_\mathrm{p}(\mathbf{y}_\mathrm{p})$ is differentiable, then the stationarity conditions are:

$$\nabla_{\mathbf{y}_\mathrm{p}} \mathcal{A}_d(\mathbf{y}_\mathrm{p}^\star) - \left(\partial_{\mathbf{y}_\mathrm{p}} \mathbf{g}_\mathrm{p}(\mathbf{y}_\mathrm{p}^\star)\right)^\mathsf{T} \boldsymbol{\mu}_\mathrm{p}^\star = \mathbf{0}_{n_\mathrm{p}}, \tag{6.71}$$

where

$$\begin{aligned} \nabla_{\mathbf{y}_\mathrm{p}} \mathcal{A}_d(\mathbf{y}_\mathrm{p}^\star) &= -(1-\alpha)\nabla_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) + \left(\frac{\partial \mathbf{w}^\star(\mathbf{y}_\mathrm{p}^\star)}{\partial \mathbf{y}_\mathrm{p}}\right)^\mathsf{T} \left(\alpha \nabla_{\mathbf{w}^\star} \mathcal{A}(\mathbf{y}_\mathrm{p}^\star) - (1-\alpha)\nabla_{\mathbf{w}^\star} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star)\right) \\ &= -(1-\alpha)\nabla_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) - \boldsymbol{\Phi}_{\mathrm{p}_e} \mathbf{S}_\lambda^{-1} \left(\frac{\alpha}{n_\mathrm{val}} \boldsymbol{\Phi}_{\mathrm{val}_e}^\mathsf{T} \left(\mathbf{y}_\mathrm{val} - \boldsymbol{\Phi}_{\mathrm{val}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p}^\star)\right)\right. \\ &\qquad \left. + (1-\alpha)\nabla_{\mathbf{w}^\star} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star)\right). \end{aligned} \tag{6.72}$$

If the detectability-risk function is $\mathcal{R}(\mathbf{y}_\mathrm{p}) = \frac{1}{2n_\mathrm{p}} ||\mathbf{y}_\mathrm{p} - \boldsymbol{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p})||_2^2$, Eq. (6.72) can be particularised for $\nabla_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) = \frac{1}{n_\mathrm{p}} \left(\mathbf{y}_\mathrm{p}^\star - \boldsymbol{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p}^\star)\right)$ and $\nabla_{\mathbf{w}^\star} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) = -\frac{1}{n_\mathrm{p}} \boldsymbol{\Phi}_{\mathrm{p}_e}^\mathsf{T} \left(\mathbf{y}_\mathrm{p}^\star - \boldsymbol{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star(\mathbf{y}_\mathrm{p}^\star)\right)$. For $\mathcal{R}(\mathbf{y}_\mathrm{p}) = \frac{1}{2n_\mathrm{p}} ||\mathbf{y}_\mathrm{p} - \mathbf{y}_{\mathrm{p}_0}||_2^2$, $\nabla_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) = \frac{1}{n_\mathrm{p}} (\mathbf{y}_\mathrm{p}^\star - \mathbf{y}_{\mathrm{p}_0})$ and $\nabla_{\mathbf{w}^\star} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) = \mathbf{0}_d$. Finally, for $\mathcal{R}(\mathbf{y}_\mathrm{p}) = \frac{1}{2d} ||\mathbf{w}^\star(\mathbf{y}_\mathrm{p}) - \mathbf{w}_\mathrm{tr}^\star||_2^2$, $\nabla_{\mathbf{y}_\mathrm{p}} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) = \mathbf{0}_{n_\mathrm{p}}$ and $\nabla_{\mathbf{w}^\star} \mathcal{R}(\mathbf{y}_\mathrm{p}^\star) = \frac{1}{d} (\mathbf{w}^\star(\mathbf{y}_\mathrm{p}^\star) - \mathbf{w}_\mathrm{tr}^\star)$.

The primal feasibility, dual feasibility, and complementary slackness conditions of the

Lagrangian (6.71) are, respectively:

$$\mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) \preceq \mathbf{0}_{m_{\mathrm{p}}}, \tag{6.73}$$

$$\boldsymbol{\mu}_{\mathrm{p}}^{\star} \succeq \mathbf{0}_{m_{\mathrm{p}}}, \tag{6.74}$$

$$\boldsymbol{\mu}_{\mathrm{p}}^{\star} \odot \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) = \mathbf{0}_{m_{\mathrm{p}}}, \tag{6.75}$$

where the operator $\odot$ denotes the Hadamard (element-wise) product. Roughly speaking, the complementary slackness conditions mean that each optimal KKT multiplier is zero unless its corresponding constraint is active (zero) at the optimum. Therefore, under optimality conditions, the poisoning labels and KKT multipliers will satisfy the following constrained system of $n_{\mathrm{p}} + m_{\mathrm{p}}$ equations, $2m_{\mathrm{p}}$ inequalities (feasible set), and $n_{\mathrm{p}} + m_{\mathrm{p}}$ variables:

$$\begin{aligned} \mathbf{f}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}, \boldsymbol{\mu}_{\mathrm{p}}^{\star}) = \begin{bmatrix} \nabla_{\mathbf{y}_{\mathrm{p}}} \mathcal{A}_d(\mathbf{y}_{\mathrm{p}}^{\star}) - \left(\partial_{\mathbf{y}_{\mathrm{p}}} \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star})\right)^{\mathsf{T}} \boldsymbol{\mu}_{\mathrm{p}}^{\star} \\ \boldsymbol{\mu}_{\mathrm{p}}^{\star} \odot \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) \end{bmatrix} = \mathbf{0}_{n_{\mathrm{p}}+m_{\mathrm{p}}} \\ \text{s.t.} \quad \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) \preceq \mathbf{0}_{m_{\mathrm{p}}}, \\ \boldsymbol{\mu}_{\mathrm{p}}^{\star} \succeq \mathbf{0}_{m_{\mathrm{p}}}, \end{aligned} \tag{6.76}$$

where $\mathbf{f}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}, \boldsymbol{\mu}_{\mathrm{p}}^{\star}) = \mathbf{0}_{n_{\mathrm{p}}+m_{\mathrm{p}}}$ is an implicit function. Note that the system of equations in (6.76) is, in general, non-linear. Therefore, it can provide a non-unique set of candidate solution for each variable $y_{\mathrm{p}_i}^{\star}, \mu_{\mathrm{p}_j}^{\star}, i = 1, \ldots, n_{\mathrm{p}}$, $j = 1, \ldots, m_{\mathrm{p}}$, subject to its constraints. Also, consider that $\boldsymbol{\mu}_{\mathrm{p}}^{\star}$ can be sparse, i.e., with few non-zero entries, due to the complementary slackness conditions.

On the Solutions to the Attacker's KKT Conditions

We shall now discuss the different solutions to the problem (6.76) depending on the characteristics of the attacker's objective function, and depending on whether the problem is unconstrained or constrained.

If we remove the constraints from the attacker's problem, and $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is convex, there is no bounded solution to the problem. If $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is concave and quadratic in $\mathbf{y}_{\mathrm{p}}$, as is the case for the detectability-risk functions analysed previously, the problem (6.76) reduces to $\nabla_{\mathbf{y}_{\mathrm{p}}} \mathcal{A}_d(\mathbf{y}_{\mathrm{p}}^{\star}) = \mathbf{0}_{n_{\mathrm{p}}}$ (see Eq. (6.72)), which leads to the system of n_{p} linear equations of n_{p}

variables:[23] $\mathbf{A}\mathbf{y}_{\mathrm{p}}^{\star} + \mathbf{b} = \mathbf{0}_{n_{\mathrm{p}}}$, where $\mathbf{A} \in \mathbb{R}^{n_{\mathrm{p}} \times n_{\mathrm{p}}}$ and $\mathbf{b} \in \mathbb{R}^{n_{\mathrm{p}}}$. From the previous system, the optimal poisoning labels are uniquely determined assuming $\mathbf{A}$ is non-singular, i.e., if $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is strongly concave. In other words, $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ only has one global maximum. There also exist solutions if $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is not strongly concave, but they are not unique in that case.

Let us set the constraints again. If $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is strongly concave[24] in $\mathbf{y}_{\mathrm{p}}$, and its feasible set is closed bounded and convex, then it is guaranteed that the problem (6.76) has a unique solution (assuming it exists). This is a direct consequence of a fundamental property of strongly concave maximisation problems—or, equivalently, strongly convex minimisation problems: any locally optimal point is also (globally) optimal and unique (Boyd and Vandenberghe, 2004). In this case, the global optimum can be attained either in the interior or at the boundary of the feasible set. If the global maximum of $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ obtained when removing the constraints lies in the interior if the feasible set, then the global maximum of the constrained problem also lies in the interior of the feasible set. Otherwise, the solution lies at the boundary of the feasible set.

If $\mathcal{A}_d(\mathbf{y}_{\mathrm{p}})$ is convex in $\mathbf{y}_{\mathrm{p}}$ and the problem is constrained, then the problem is non-convex, and the KKT conditions reduce to the stationarity conditions (6.71) and the primal feasibility conditions:

$$\mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) = \mathbf{0}_{m_{\mathrm{p}}}, \tag{6.77}$$

Therefore, under optimality conditions, the poisoning labels and KKT multipliers will satisfy the following system of $n_{\mathrm{p}} + m_{\mathrm{p}}$ equations and $n_{\mathrm{p}} + m_{\mathrm{p}}$ variables:

$$\mathbf{f}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}, \boldsymbol{\mu}_{\mathrm{p}}^{\star}) = \begin{bmatrix} \nabla_{\mathbf{y}_{\mathrm{p}}} \mathcal{A}_d(\mathbf{y}_{\mathrm{p}}^{\star}) - \left(\partial_{\mathbf{y}_{\mathrm{p}}} \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star})\right)^{\mathsf{T}} \boldsymbol{\mu}_{\mathrm{p}}^{\star} \\ \mathbf{g}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}) \end{bmatrix} = \mathbf{0}_{n_{\mathrm{p}}+m_{\mathrm{p}}} \tag{6.78}$$

where $\mathbf{f}_{\mathrm{p}}(\mathbf{y}_{\mathrm{p}}^{\star}, \boldsymbol{\mu}_{\mathrm{p}}^{\star}) = \mathbf{0}_{n_{\mathrm{p}}+m_{\mathrm{p}}}$ is again an implicit function. This corresponds to the standard method of Lagrange multipliers.

From Bauer's maximum principle (Bauer, 1958), we know that the poisoning labels

[23] This can be easily seen by noticing that $\mathbf{w}^{\star}(\mathbf{y}_{\mathrm{p}})$ is an affine function of $\mathbf{y}_{\mathrm{p}}$.

[24] A function $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is strongly concave on its feasible set $\mathcal{F}$, if there exists an $m > 0$ such that the negative Hessian $-\nabla_{\mathbf{x}}^2 f(\mathbf{x}) \succeq m\mathbf{I}_n$. Furthermore, the maximum eigenvalue of $-\nabla_{\mathbf{x}}^2 f(\mathbf{x})$, which is a continuous function of $\mathbf{x}$ on $\mathcal{F}$, is bounded above on $\mathcal{F}$, i.e., there exists a constant d such that $-\nabla_{\mathbf{x}}^2 f(\mathbf{x}) \preceq d\mathbf{I}_n$. The ratio $\kappa = d/m$ is thus an upper bound on the condition number of the matrix $-\nabla_{\mathbf{x}}^2 f(\mathbf{x})$, i.e., the ratio of its largest eigenvalue to its smallest eigenvalue (Boyd and Vandenberghe, 2004). For quadratic functions, strict concavity implies strong concavity and vice versa. Hence, a quadratic function is strongly concave if and only its Hessian matrix is negative definite. In this case, the constant m corresponds to the smallest eigenvalue of $-\nabla_{\mathbf{x}}^2 f(\mathbf{x})$ (Boyd and Vandenberghe, 2004).

that are solutions to the problem (6.78) will lie at the boundary of the attacker's feasible set. For instance, if the feasible set is a box, there can be up to 2^{n_p} combinations of optimal poisoning labels, as explained in Subsect. 6.5.1. Note that, if $\frac{1}{2}(\mathbf{y}_{\mathrm{p}_{\max}} - \mathbf{y}_{\mathrm{p}_{\min}}) = \varepsilon_\infty \mathbf{1}_{n_\mathrm{p}}$ (ℓ_∞-norm), then the optimal poisoning labels can only take two possible values: (although the number of combinations is still 2^{n_p}): $\mathbf{y}_\mathrm{p}^\star \in \{y_{\mathrm{p}_{\min}}, y_{\mathrm{p}_{\max}}\}^{n_\mathrm{p}}$. From a detectability perspective, it may not be desirable that the poisoning labels lie in a reduced set of locations at the boundary.

Increasing the Diversity of Poisoning Points

If the attacker aims to increase the diversity of the optimal poisoning labels, a possible strategy can be to make $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$ not concave or convex. A feasible option is then to increase the polynomial degree of the objective function, in particular by increasing the polynomial degree of the detectability-risk function. In this case, the problem in (6.76) would consist of a constrained system of multivariate polynomial equations in $\mathbf{y}_\mathrm{p}^\star$ that could increase the number of solutions (local optima of $\mathcal{A}_d(\mathbf{y}_\mathrm{p})$) for the optimal poisoning labels. Moreover, if the system of equations in (6.76) is non-degenerate and satisfy its constraints, due to the implicit function theorem, we know that its solution space naturally has the structure of a smooth manifold. These solutions are called algebraic variety[25] in the algebraic geometry community,[26] or manifold in the ML community. Namely, an algebraic variety is the set of common zeros to a collection of multivariate polynomials. Without diving into technical details, the take-away is that this variety can help obtain more diverse poisoning points.

Along this lines, in Subsect. 6.7.4 we will justify our detectability-risk function proposed in Ch. 5. Let us first describe the last (but not least) challenge stated at the beginning of this section: order of magnitude of the effectiveness and detectability functions.

[25] In our case, the algebraic variety is defined as the set of poisoning labels satisfying a system of multivariate polynomial equations. According to Hilbert's basis theorem, a finite number of equations suffices. The analysis of algebraic varieties of poisoning labels that can be more difficult to defend against by state-of-the-art defences is left as future work.

[26] Note that the zeros of multivariate polynomials do not typically form a discrete set. In general, multivariate polynomials can have infinitely many roots. Consider the example of $f(x, y) = y - x^2$, where $x, y, f(x, y) \in \mathbb{R}$. The points in the plane which satisfy the equation $f(x, y) = 0$ describe a parabola. That is, the fundamental theorem of algebra typically does not hold true for multivariate polynomials, and it is generalised by the weak form of Hilbert's Nullstellensatz, and by Bézout's theorem when considering a system of multivariate polynomial equations, which we will not describe here.

6.7.3 Zeroth-Order Challenge: Order of Magnitude of Objective Functions

The zeroth-order challenge arises when the order of magnitude of the effectiveness objective is very different from the order of magnitude of the detectability risk objective, as explained in Ch. 5. This prevents a smooth transition from inliers to outliers when crafting the poisoning points, and hinders the interpretability for α, i.e., the stealth of the attack.

Let us assume that the functions $\mathcal{A}$ and $\mathcal{R}$ are both convex, as shown previously. If the order of magnitude of the effectiveness objective is much larger than the one of the detectability risk objective, i.e., $\frac{\mathcal{A}}{\mathcal{R}} \gg 1$ in general then, according to Eq. (6.45), $\mathcal{A}_d \approx \mathcal{A}$ when α is not close to 0. In that case, $\mathcal{A}_d$ becomes convex and the optimal poisoning points lie at the boundary of the attacker's feasible set. Conversely, if the order of magnitude of the effectiveness objective is much smaller than the one of the detectability risk objective, i.e., $\frac{\mathcal{A}}{\mathcal{R}} \ll 1$ in general then, according to Eq. (6.45), $\mathcal{A}_d \approx -\mathcal{R}$ when α is not close to 1. Thus $\mathcal{A}_d$ becomes concave and the optimal poisoning points lie in the local maximum of the detectability-risk function. This is not a desirable effect for the sake of the diversity of the poisoning points, as explained in Subsect. 6.7.1. Moreover, this effect does not allow to sweep different values of α accordingly, which does not allow to interpret different trade-offs between effectiveness and detectability. For this reason, we proposed the normalisation algorithm for the objective functions, described in Ch. 5, Subsect. 5.3.2.

6.7.4 Proposed Detectability-Risk Function

To alleviate the second-order and first-order challenges, in Ch. 5 we proposed the following detectability-risk function (Eq. (5.6)):

$$\begin{aligned}
\mathcal{R} &= \frac{1}{2n_{\mathrm{p}}}||\mathbf{y}_{\mathrm{p}} - \mathbf{y}_{\mathrm{target}_1}||_2^2 \frac{1}{2n_{\mathrm{p}}}||\mathbf{y}_{\mathrm{p}} - \mathbf{y}_{\mathrm{target}_2}||_2^2 \\
&= \frac{1}{4n_{\mathrm{p}}^2}||\mathbf{y}_{\mathrm{p}} - \mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}^\star_{\mathrm{target}_1}||_2^2||\mathbf{y}_{\mathrm{p}} - \mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}^\star_{\mathrm{target}_2}||_2^2 \\
&= \frac{1}{4n_{\mathrm{p}}^2}\left(||\mathbf{y}_{\mathrm{p}}||_2^2 - 2\mathbf{y}_{\mathrm{p}}^\mathsf{T}\mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}^\star_{\mathrm{target}_1} + ||\mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}^\star_{\mathrm{target}_1}||_2^2\right) \\
&\qquad \left(||\mathbf{y}_{\mathrm{p}}||_2^2 - 2\mathbf{y}_{\mathrm{p}}^\mathsf{T}\mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}^\star_{\mathrm{target}_2} + ||\mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}^\star_{\mathrm{target}_2}||_2^2\right),
\end{aligned} \tag{6.79}$$

which is a quartic function in $\mathbf{y}_{\mathrm{p}}$, where $\mathbf{y}_{\mathrm{target}_1} \neq \mathbf{y}_{\mathrm{target}_2}$, and where $\mathbf{w}^\star_{\mathrm{target}_1}$ and $\mathbf{w}^\star_{\mathrm{target}_2}$ follow a distribution similar (within a certain confidence interval) to the one of the clean

parameters, $\mathbf{w}^\star_{\text{tr}}$. $\mathcal{R}(\mathbf{y}_\text{p})$ is not convex or concave in $\mathbf{y}_\text{p}$ in general,[27] since it is the product of two convex quadratic functions in $\mathbf{y}_\text{p}$, which are both increasing and decreasing for certain values of $\mathbf{y}_\text{p}$. In particular, the two local maxima of $-\mathcal{R}(\mathbf{y}_\text{p})$ are located at $\mathbf{y}^\star_{\text{p}1} = \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_1}$ and $\mathbf{y}^\star_{\text{p}2} = \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_2}$, respectively. The hyper-Hessian of $\mathcal{R}(\mathbf{y}_\text{p})$ in $\mathbf{y}_\text{p}$ is:

$$\begin{aligned}\nabla^2_{\mathbf{y}_\text{p}}\mathcal{R}(\mathbf{y}_\text{p}) = \frac{1}{2n_\text{p}^2}\left(||\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_1}||_2^2 + ||\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_2}||_2^2\right)\mathbf{I}_{n_\text{p}} \\ + \frac{1}{n_\text{p}^2}\Big((\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_1})(\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_2})^\mathsf{T} \\ +(\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_2})(\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_1})^\mathsf{T}\Big),\end{aligned} \tag{6.80}$$

which is clearly indefinite in general—except when $\mathbf{w}^\star_{\text{target}_1} = \mathbf{w}^\star_{\text{target}_2}$, in which case, the hyper-Hessian would be positive semidefinite and therefore $\mathcal{R}(\mathbf{y}_\text{p})$ would be convex—and hence $\mathcal{R}(\mathbf{y}_\text{p})$ is not convex or concave in $\mathbf{y}_\text{p}$. Since the Loewner order is defined by the convex cone of positive semidefinite matrices, unless $\mathbf{w}^\star_{\text{target}_1} = \mathbf{w}^\star_{\text{target}_2}$ or $\alpha = 1$, the condition for convexity for $\mathcal{A}_d(\mathbf{y}_\text{p})$ does not hold true in general, so that $\mathcal{A}_d(\mathbf{y}_\text{p})$ is not convex or concave in general. Thus, it is less probable that the optimal poisoning labels lie at the boundary of the attacker's feasible set, which can increase the difficulty to detect them.

In addition, the hypergradient of $\mathcal{R}(\mathbf{y}_\text{p})$ is:

$$\begin{aligned}\nabla_{\mathbf{y}_\text{p}}\mathcal{R}(\mathbf{y}_\text{p}) = \frac{1}{2n_\text{p}^2}\Big(||\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_1}||_2^2(\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_2}) \\ +||\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_2}||_2^2(\mathbf{y}_\text{p} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}^\star_{\text{target}_1})\Big),\end{aligned} \tag{6.81}$$

noticing that $\nabla_{\mathbf{w}^\star}\mathcal{R}(\mathbf{y}_\text{p}) = \mathbf{0}_d$. Thus, each component of the hypergradient is cubic in $\mathbf{y}_\text{p}$. Consequently, without considering the feasible set, and under stationarity conditions, combining the hypergradient in Eq. (6.81) and the hypergradient of the effectiveness objective (which is linear in $\mathbf{y}_\text{p}$) leads to a system of n_p n_p-variate cubic equations. Effectively, this is an algebraic variety that can increase the number of combinations of optimal poisoning labels.

Furthermore, because $\mathbf{w}^\star_{\text{target}_1}$ and $\mathbf{w}^\star_{\text{target}_2}$ do not depend on the poisoning points, $\mathbf{y}_{\text{target}_1}$ and $\mathbf{y}_{\text{target}_2}$ remain constant during the optimisation algorithm of the poisoning points. In other words, the reference for the detectability-risk function does not change regardless of the number and value of the poisoning points, which imitates a defender who

[27] Note that the detectability-risk function is also not concave or concave in $\mathbf{\Phi}_\text{p}$ in general.

has access to a clean validation set to be compared with the poisoned training set.

In conclusion, an attacker's objective function of, at least, third polynomial degree can increase the diversity of the optimal poisoning points, and therefore the difficulty to detect them, as empirically shown in Ch. 5. If the effectiveness objective is convex and quadratic, and the detectability-risk function is non-negative, then a quartic detectability-risk function can provide diversity for the optimal poisoning points.

6.8 Conclusions

In this chapter, we have analysed the behaviour of the attacker's objective in optimal indiscriminate attacks against ridge regression, to understand the limitations discussed in Ch. 5. We have shown that, without considering soft detectability constraints, the attacker's objective is convex in the poisoning labels, but non-convex in the poisoning basis functions, and therefore it is non-convex when combining both sets of variables.

In particular, given the stationarity conditions for the basis functions, we have conjectured that most of the critical points of the attacker's objective are saddle points. This requires heuristics, e.g., normalisation of hypergradients, to help escape the saddle points and speed up the optimisation process.

With regard to the asymptotic behaviour of the attacker's objective, we theoretically determined the intuitive result that the attacker is generally unbounded if the poisoning basis functions and/or labels are unbounded. However, regularising the model can prevent the attacker from being unbounded, which is coherent with the results obtained in classification settings in Ch. 3.

We have also shown the equivalence between the optimal indiscriminate attack and CQP (if only the poisoning labels are optimised), and obtained some bounds on the rank of the hyper-Hessian matrix—Hessian matrix of the attacker's objective function: under certain assumptions, the hyper-Hessian is non-singular if and only if all the poisoning points are linearly independent in the basis functions (or feature) space. Therefore, optimising more poisoning points than the number of basis functions may be redundant.

When considering soft detectability constraints, we have shown that the attacker's objective's convexity can be brittle: because the attacker's objective is a sum of a convex objective—effectiveness—and a negative convex (concave) objective—detectability—the

objective function can undesirably transition from convexity to concavity, depending on the values of their respective hyper-Hessians and the hyperparameter that controls their trade-off, α. We also defined the attacker's multiobjective condition number, which is the ratio between the largest eigenvalue of the hyper-Hessian of the effectiveness objective, and the smallest eigenvalue of the hyper-Hessian of the detectability objective. In practice, a very large value of the condition number leads to an objective function predominantly convex—the poisoning labels lie at the boundary of the feasible set. In contrast, a very low value of the condition number means that the objective is predominantly concave—the poisoning labels lie in the local minimum of the detectability-risk function, if the minimum is in the interior of the feasible set, or at its boundary otherwise. We also obtained a relationship between α and the condition number that determines the concavity of the objective.

Finally, the lessons learnt allowed us to develop the attacker's KKT conditions, and to discuss the different solutions depending on the convexity of the attacker's objective. We also justified the detectability-risk function proposed in Ch. 5, based on a product of two convex functions, which can provide more diversity of poisoning points—and potentially increasing the difficulty to detect them. If the effectiveness objective is convex and quadratic, and the detectability-risk function is non-negative, then a quartic detectability-risk function can provide diversity for the optimal poisoning points.

In Ch. 7 we will theoretically analyse another observation obtained from the results in Ch. 5: state-of-the-art defences such as TRIM (Jagielski et al., 2018) and SEVER (Diakonikolas et al., 2019) worsen their performance from a certain ratio of poisoning points.

Chapter 7

Poisonability of Regression Models

Synopsis

We challenge the underlying assumption of most defences in regression settings—i.e., the error of the poisoning points is larger than the error of the clean points—and introduce the concept of "poisonability" of models and datasets. The poisonability allows us to find the number of poisoning points required so that the mean error of the clean points matches the tolerable mean error of the poisoning points on the poisoned model. In particular, we study the poisonability of linear regression models according to the characteristics of the clean training points and poisoning points.

7.1 Introduction

In regression settings, state-of-the-art data sanitisation techniques and defences against data poisoning attacks assume that the error of the poisoning points on the *poisoned model* is larger than the error of the clean points on the *poisoned model* (Jagielski et al., 2018; Diakonikolas et al., 2019). These works typically derive methods based on rejecting points whose error or the norm of their corresponding gradients are large. This assumption is reasonable when the fraction of poisoning points is low and the points are located far from the clean distribution, as we observed in Ch. 5. However, this is not always the

case and does not represent advanced attack scenarios where the attacker aims to evade detection.

Hence, in this chapter, we shall answer the following research question stated in Ch. 1:

- **RQ6**: Can we define how "poisonable" a particular dataset and model are? Specifically, does the assumption that the error of the poisoning points is larger than the error of the clean points always hold true?

In this context, we introduce the concept of *"poisonability" of models and datasets*. This measure aims to determine the number of poisoning points needed so that a certain tolerable mean error of the poisoning points on the poisoned model matches the mean error of the clean points on the poisoned model. We also introduce the concept of breaking points of a model and dataset, which effectively shows that the previous condition can happen for different ratios of poisoning points. These measures can also help understand why certain defences fail when the ratio of poisoning points in the training set is large. We then analyse the poisonability of linear regression models.

7.2 Related Work

Well-established pioneer related works were developed in the statistics literature during the decades of 1960s, 1970s, and 1980s. In particular, Hampel (1971) proposed a formal definition of the breakdown point of a parametric model, extending a definition of Hodges (1967): "the breakdown point is the smallest percentage of contaminated data that can cause the estimator to take on arbitrarily large aberrant values. In the case of least squares, the breakdown point is zero" (Rousseeuw, 1984). In the subfield of *regression diagnostics*, Belsley et al. (1980) described an explicit relationship between the clean and corrupted parameters of a linear regression model when a single data point is removed.

Another related line of research is certified defences against poisoning attacks, which guarantee that deleting a fixed number of instances from the training data will not change a model's prediction (Steinhardt et al., 2017; Levine and Feizi, 2021; Jia et al., 2022; Wang et al., 2022; Hammoudeh and Lowd, 2023). These methods can be viewed as upper bounding the training data's group influence—although very coarsely. Most certified poisoning defences achieve their bounds by leveraging "tricks" associated with particular model architectures (e.g., nearest-neighbours learners (Jia et al., 2022) and ensembles

(Levine and Feizi, 2021; Wang et al., 2022; Rezaei et al., 2023) as opposed to a detailed analysis of a prediction's stability (Hammoudeh and Lowd, 2023). For instance, given some assumptions on the learning procedure, Steinhardt et al. (2017) propose a method of finding approximate upper bounds on the effectiveness of data poisoning attacks on the learner's loss, and introduce an attack that nearly achieves that bound in two settings: a data-independent and data-dependent defence. In summary, certified methods investigate guarantees in terms of possible bounds or variations of the loss function and/or error when the training data is poisoned, whereas we aim to understand how the model combines the clean and poisoning distributions and their relationship and dominance in terms of their respective errors.

Lu et al. (2023) introduce the notion of model poisoning reachability in model-targeted data poisoning. Under stationarity conditions and certain convexity conditions, they theoretically determine the poisoning ratio that any data poisoning attack has to satisfy to attain some given target parameters. One limitation of their work is its focus on achieving specific target parameters, which may not always be available or necessary. This study builds upon the work of Koh et al. (2022) which, in the case of logistic regression and binary SVM, essentially shows that only two different poisoning points—copied potentially many times—are needed to attain some target poisoned parameters. Even if these target parameters are attainable, the errors of the clean points and poisoning points do not necessarily match (for instance, the target parameters may be very similar to the clean parameters). In contrast, we ask the following research question: under what conditions the clean and poisoned distributions match? And we define that condition in terms of their respective mean errors on the poisoned model—under stationarity conditions. We also ask the research question: what is the relationship between the clean and poisoned parameters?

7.3 On the Poisonability of Models and Datasets

As in Ch. 4, 5, and 6, we consider parametric regression models. Given the input space $\mathcal{X} \subseteq \mathbb{R}^m$ and the continuous label space, $\mathcal{Y} \subseteq \mathbb{R}$, the learner aims to estimate the mapping $f : \mathcal{X} \to \mathcal{Y}$. Given the clean training set $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}_{\text{tr}_i}, y_{\text{tr}_i})\}_{i=1}^{n_{\text{tr}}} = (\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}})$ with n_{tr} IID samples drawn from the underlying probability distribution $p(\mathcal{X}, \mathcal{Y})$, we can estimate f with a model $\mathcal{M}$ trained by minimising an objective function $\mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w})$ w.r.t. its parameters,

$\mathbf{w} \in \mathbb{R}^d$. The parameters learnt under the clean training set, $\mathcal{D}_{\text{tr}}$, are therefore $\mathbf{w}^\star_{\text{tr}}$.

In a poisoning attack, the adversary aims to inject a set of n_{p} poisoning data points, $\mathcal{D}_{\text{p}} = \{(\mathbf{x}_{\text{p}_k}, y_{\text{p}_k})\}_{k=1}^{n_{\text{p}}} = (\mathbf{X}_{\text{p}}, \mathbf{y}_{\text{p}})$, in the training set. Thus, under the poisoning attack, the learner minimises the objective function $\mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}, \mathbf{w})$. The parameters learnt under the poisoned training set, $\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}$, are thus $\mathbf{w}^\star$.

In this chapter, we challenge the assumption of previous defences in regression settings (Jagielski et al., 2018; Diakonikolas et al., 2019) and show, theoretically, that the mean error of the clean points can be larger than the mean error of the poisoning points on the poisoned model, if the number of poisoning points is large enough. Namely, the model is sufficiently poisoned that it is no longer a reliable reference for the classical assumption of standard defences.

As in previous chapters, we also assume a worst-case scenario, i.e., the attacker has perfect knowledge about the model, feature representation and the data of the defender. This provides a lower bound on the performance when the model is under attack.

Definition 7.1 (Breaking point)**.** *We define a breaking point* $n^\star_{\text{p}_i} \in \mathbb{N}_{\geq 1}$ *of a target model* $\mathcal{M}$ *trained on a poisoned training set* $\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}$ *as the number of poisoning points required so that the mean error of* $\mathcal{D}_{\text{tr}}$ *on* $\mathcal{M}$ *matches the mean error of* $\mathcal{D}_{\text{p}}$ *on* $\mathcal{M}$*.*

Definition 7.2 (Poisonability problem)**.** *We define the poisonability problem as the problem of finding the set of* breaking points *(which* may not be unique*),* $\mathcal{N}^\star_{\text{break}} = \{n^\star_{\text{p}_i}\}_{i=1}^{|\mathcal{N}^\star_{\text{break}}|}$*, of a target model* $\mathcal{M}$*.*

We will mathematically define the breaking points of a model in Subsect. 7.3.1. Note that $\mathcal{D}_{\text{tr}}$ is fixed, and the variable to explore is $\mathcal{D}_{\text{p}}$. Clearly, the problem depends on the definition of $\mathcal{D}_{\text{p}}$, i.e., its features and labels. We shall discuss this definition in Subsect. .7.3.3.

In particular, computing the minimum and maximum values of the breaking points, $n^\star_{\text{p}_{\text{inf}}} = \min_i \mathcal{N}_{\text{break}}$ and $n^\star_{\text{p}_{\text{sup}}} = \max_i \mathcal{N}_{\text{break}}$, respectively, can be of special interest, because they can give a sense of how "poisonable" a particular dataset and model are. If $n^\star_{\text{p}_{\text{inf}}}$ is large, an attacker needs to inject a large number of points to break the assumption of the defence. Also, $n^\star_{\text{p}_{\text{sup}}}$ gives a sense of the breaking point from which the model always emphasises more the poisoning distribution.

7.3.1 Formulation of the Problem

Mathematically, the breaking points of a model $\mathcal{M}$ satisfy the following integer root-finding problem:

$$\begin{aligned} \text{find} \quad & \mathcal{N}^\star_{\text{break}} = \{n^\star_{\text{p}_i}\}_{i=1}^{|\mathcal{N}^\star_{\text{break}}|} \\ \text{s.t.} \quad & p(n^\star_{\text{p}_i}, \xi_\text{p}) = 0, \\ & n^\star_{\text{p}_i} \in \mathbb{N}_{\geq 1}, \end{aligned} \tag{7.1}$$

where

$$p(n_\text{p}, \xi_\text{p}) = \mathcal{L}(\mathcal{D}_\text{p}(n_\text{p}), \mathbf{w}^\star(n_\text{p})) - \xi_\text{p}\mathcal{L}(\mathcal{D}_\text{tr}, \mathbf{w}^\star(n_\text{p})) \tag{7.2}$$

is the objective function that characterises the poisonability problem. We denote the dependency of $\mathcal{D}_\text{p}$ and $\mathbf{w}^\star_\text{p}$ on n_p through $\mathcal{D}_\text{p}(n_\text{p})$ and $\mathbf{w}^\star_\text{p}(n_\text{p})$, respectively. In Eq. (7.2), $\mathcal{L} \geq 0$ is the loss function of the model and $\xi_\text{p} > 0$ is the *poisonability intolerance* (let $\xi_\text{p}^{-1} > 0$ be the *poisonability tolerance*). The poisonability intolerance is a hyperparameter that quantifies how close we allow the mean error of the poisoning points to be with respect to the mean error of the clean points on the poisoned model. It allows us to define poisonability even when the clean and poisoning mean errors match for a very low or a very large number of poisoning points. Note that for $\xi_\text{p} = 1$, both errors match.

7.3.2 Solving the Poisonability Problem

Without additional assumptions, a brute-force approach to solving (7.1) is to explore all possible values of n_p. Under optimality conditions, in worst-case scenarios, this is equivalent to, for each possible n_{p_i}, solving the optimal attack problem, which is a bilevel optimisation problem of the form:

$$\begin{aligned} \max_{\mathcal{D}_\text{p}(n_{\text{p}_i})} \quad & \mathcal{A}(\mathbf{w}^\star(n_{\text{p}_i})) \\ \text{s.t.} \quad & \mathbf{w}^\star(n_{\text{p}_i}) \in \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_\text{tr} \cup \mathcal{D}_\text{p}(n_{\text{p}_i}), \mathbf{w}), \\ & \mathcal{D}_\text{p}(n_{\text{p}_i}) \in \mathcal{F}_{\mathcal{D}_\text{p}}, \end{aligned} \tag{7.3}$$

where $\mathcal{A}$ is the attacker's objective and $\mathcal{F}_{\mathcal{D}_\text{p}}$ is the feasible set of the attacker.

Solving (7.3) is computationally challenging, and it depends on the definition of the attacker's feasible set. Moreover, combining (7.3) with (7.1) is equivalent to solving a (possibly) infinite number of bilevel optimisation problems of potentially increasing

complexity for $n_{p_{i+1}} > n_{p_i}$, since it requires, for each n_{p_i}: (i) computing the new optimal poisoning points via bilevel optimisation and (ii) retraining the model from scratch for each new poisoned training set. In addition, the solution may vary depending on how the previous poisoning points are considered, and how n_{p_i} increases: computing from scratch the whole new set of poisoning for each n_{p_i} (which implies that each set of n_{p_i} points is independent of the others) is not necessarily the same as the online assumption of considering the previous n_{p_i} poisoning points fixed, and only computing the new $n_{\mathrm{p}_{i+1}} - n_{\mathrm{p}_i}$ points each time (which implies that each new set of n_{p_i} points is dependent upon the previous ones). Also, if $n_{\mathrm{p}_{i+1}} = n_{\mathrm{p}_i} + v_{\mathrm{p}}$, where v_{p} is the number of poisoning points computed at the same time, the solution may change depending on v_{p}. Moreover, this approach does not allow to systematically study the robustness of the model to fixed characteristics of the poisoning points, since the optimal poisoning points may change for each n_{p_i}, so it is more difficult to draw general conclusions and it would be still an approximation.

Instead of studying a particular type of attack formulation or attacker's goal, we aim to study the poisonability of a model and dataset following a more holistic approach, i.e., depending on the characteristics of the clean training data, poisoning points, and target model.

Therefore, given a fixed target model $\mathcal{M}$, a fixed clean training set $\mathcal{D}_{\mathrm{tr}}$, and a variable poisoning set $\mathcal{D}_{\mathrm{p}}(n_{\mathrm{p}}) = \{(\mathbf{x}_{\mathrm{p}_i}, y_{\mathrm{p}_i})\}_{i=1}^{n_{\mathrm{p}}}$, the steps to solve the poisonability problem are:

1. Learn the clean parameters $\mathcal{D}_{\mathrm{tr}} \rightarrow \mathbf{w}^{\star}_{\mathrm{tr}}(\mathcal{D}_{\mathrm{tr}})$. For a single poisoning point $(\mathbf{x}_{\mathrm{p}}, y_{\mathrm{p}})$, learn the poisoned parameters $(\mathcal{D}_{\mathrm{tr}}, \mathbf{x}_{\mathrm{p}}, y_{\mathrm{p}}) \rightarrow \mathbf{w}^{\star}(\mathcal{D}_{\mathrm{tr}}, \mathbf{x}_{\mathrm{p}}, y_{\mathrm{p}})$, and derive a mapping between the clean and poisoned parameters: $\mathbf{w}^{\star}(\mathcal{D}_{\mathrm{tr}}, \mathbf{x}_{\mathrm{p}}, y_{\mathrm{p}}, \mathbf{w}^{\star}_{\mathrm{tr}}(\mathcal{D}_{\mathrm{tr}}))$.

2. Under certain assumptions for a set of poisoning points $\mathcal{D}_{\mathrm{p}}(n_{\mathrm{p}})$, generalise the previous mapping for $\mathcal{D}_{\mathrm{p}}(n_{\mathrm{p}})$, either through the mapping for a single poisoning point: $\{\mathbf{w}^{\star}_i(\mathcal{D}_{\mathrm{tr}}, \mathbf{x}_{\mathrm{p}_i}, y_{\mathrm{p}_i}, \mathbf{w}^{\star}_{\mathrm{tr}}(\mathcal{D}_{\mathrm{tr}}))\}_{i=1}^{n_{\mathrm{p}}} \rightarrow \mathbf{w}^{\star}(\mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{p}}(n_{\mathrm{p}}), \mathbf{w}^{\star}_{\mathrm{tr}}(\mathcal{D}_{\mathrm{tr}}))$; or directly from the clean parameters: $\mathbf{w}^{\star}(\mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{p}}(n_{\mathrm{p}}), \mathbf{w}^{\star}_{\mathrm{tr}}(\mathcal{D}_{\mathrm{tr}}))$, $\forall n_{\mathrm{p}}$.

3. Based on the mapping $\mathbf{w}^{\star}(\mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{p}}(n_{\mathrm{p}}), \mathbf{w}^{\star}_{\mathrm{tr}}(\mathcal{D}_{\mathrm{tr}}))$, construct the objective function $p(n_{\mathrm{p}}, \xi_{\mathrm{p}})$ (7.2) for a particular poisonability intolerance ξ_{p}.

4. Solve the root-finding problem (7.1) to find the breaking points $\mathcal{N}_{\mathrm{break}}$.

Note that, without additional assumptions, finding the mapping $\mathbf{w}^\star(\mathcal{D}_{\text{tr}}, \mathcal{D}_{\text{p}}(n_{\text{p}}), \mathbf{w}^\star_{\text{tr}}(\mathcal{D}_{\text{tr}}))$ is computationally challenging, because it requires to enumerate all possible values of $\mathcal{D}_{\text{p}}(n_{\text{p}})$, which is an infinite set. Even assuming that the sample space is finite, it can grow exponentially with n_{p}. For instance, assuming unidimensional binary features and binary labels, the number of combinations is $4^{n_{\text{p}}}$, namely $\mathcal{O}(2^{n_{\text{p}}})$. Hence, further assumptions for the sample space over $\mathcal{D}_{\text{p}}(n_{\text{p}})$ are needed to make the problem tractable. In essence, this means assuming a certain distribution for the poisoning points.

Beyond the poisonability problem, finding a mapping between the clean and poisoned parameters can also help understand the consequences of the attack and define standards of performance degradation with respect to a clean reference model. For example, knowing the value of the parameters of a poisoned model's parameter can itself be meaningless, but knowing the *transformation* of the model's parameters after the attack can provide more information about the attack's causality, as well as help define the desired tolerance of the model.

7.3.3 Approximation to the Problem

In offline learning settings, one can choose a finite sample space over $\mathcal{D}_{\text{p}}$, $\mathcal{D}_{\text{p}_{\text{prot}}} = \{(\mathbf{x}_{\text{p}_{\text{prot}_i}}, y_{\text{p}_{\text{prot}_i}})\}_{i=1}^{c_{\text{p}}} \in \mathcal{D}_{\text{p}}$, which we call *poisoning prototypes*, representative of a particular poisoning attack. Moreover, assuming a stationary attacker that does not change the distribution of the poisoning points, its number of different elements c_{p} is constant over n_{p}, i.e., $\mathcal{O}(c_{\text{p}})$.[1] That is to say, the distribution of the poisoning points does not change over n_{p} and can be modelled from the poisoning prototypes. For instance, one can model the attack distribution $f_{\mathcal{A}}(\mathbf{x}_{\text{p}}, y_{\text{p}})$ as a weighted sum of Dirac delta functions located at the prototypes:

$$f_{\mathcal{A}}(\mathbf{x}_{\text{p}}, y_{\text{p}}) = \frac{1}{c_{\text{p}}} \sum_{i=1}^{c_{\text{p}}} \delta\left(\mathbf{x}_{\text{p}} - \mathbf{x}_{\text{p}_{\text{prot}_i}}, y_{\text{p}} - y_{\text{p}_{\text{prot}_i}}\right) \tag{7.4}$$

As a first approximation, as we shall see in Sect. 7.4, one can assume that $\mathcal{D}_{\text{p}}(n_{\text{p}}) = \{(\mathbf{x}_{\text{p}}, y_{\text{p}})\}_{i=1}^{n_{\text{p}}}$, i.e., $\mathcal{D}_{\text{p}}(n_{\text{p}})$ is a set of n_{p} identical copies of $\mathcal{D}_{\text{p}_{\text{prot}}} = (\mathbf{x}_{\text{p}}, y_{\text{p}})$. This corresponds to a rank-1 attack.

To calculate each $n^\star_{\text{p}}$, i.e., the breaking point, which is a positive integer variable, we shall proceed with a continuous relaxation of the problem. Hence, we define the real

[1] This can be a reasonable approximation if the attack is optimal.

variables greater than or equal to one: $\hat{n}_{\mathrm{p}}^{\star}, \hat{n}_{\mathrm{p}} \in \mathbb{R}_{\geq 1}$, so that $n_{\mathrm{p}}^{\star} = \lceil \hat{n}_{\mathrm{p}}^{\star} \rceil$ and $n_{\mathrm{p}} = \lceil \hat{n}_{\mathrm{p}} \rceil$. This relaxation allows to solve the problem using standard root-finding or optimisation techniques. The solution quality of the relaxed problem is at least as good as that of the integer problem, because any integer problem solution would also be a valid continuous problem solution. For example, in a minimisation problem, the relaxed problem has a value smaller than or equal to that of the original problem. Thus, the relaxation provides an optimistic bound on the integer problem solution. Under this relaxation, it is possible to find several breaking points that indeed correspond to only one point, e.g., if $\lceil \hat{n}_{\mathrm{p}_i}^{\star} \rceil = \lceil \hat{n}_{\mathrm{p}_j}^{\star} \rceil$, $i \neq j$.

After defining the formulation of the poisonability problem and a possible approach to solving it, in the next section we will focus on the poisonability of linear regression models, given their relevance in this thesis.

7.4 On the Poisonability of Linear Regression

In this section we study the poisonability of linear regression. That is, we aim to determine the breaking points of linear regression. As in Ch. 6, the following derivation is also applicable to non-linear parametric regression, as long as the model is linear in the parameters, e.g., polynomial regression, logarithmic regression or exponential regression. We also extend the class of models by considering linear combinations of $d-1$ fixed non-linear functions of the input. For more details on our notation, see Sect. 6.2. Also note that our analysis is also applicable to non-poisoned pre-trained DNNs, e.g., in transfer learning settings,[2] where the parameters at the output layer $\mathbf{w}$ are trained (and potentially poisoned) for a regression task and the rest of the parameters are frozen. In that case, $\boldsymbol{\Phi}$ corresponds to the output of the penultimate layer. If the parameters at the output layer were initialised with values $\mathbf{w}_{\text{init}} \neq \mathbf{0}_d$,[3] the output would have to be recomputed as $y' = y - \boldsymbol{\phi}_e^{\mathsf{T}} \mathbf{w}_{\text{init}}$ for our analysis. On the other hand, our analysis is also applicable to least-squares classification, e.g., the adaptive linear neuron (ADALINE) algorithm.

Hence, let us consider a poisoned linear regression model with poisoned parameters $\mathbf{w}^{\star} \in \mathbb{R}^d$. In the next subsections, we shall analyse the poisonability of both non-regularised

[2] For instance, this could be the case of a convolutional neural network (CNN) pre-trained on ImageNet and applied to regression tasks like estimating age from a face, or the location of facial keypoints.

[3] We denote a column vector of zeros as $\mathbf{0}_d \in \mathbb{R}^d$.

and L_2-regularised linear regression, a.k.a. ridge regression or Tikhonov regularisation, given its common use by ML researchers and practitioners. These models also allow us to derive closed-form solutions. Hence, the loss function $\mathcal{L}$ of the defender is:

$$\mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}, \mathbf{w}) = \frac{1}{2(n_{\text{tr}} + n_{\text{p}})} \left(||\mathbf{y}_{\text{tr}} - \mathbf{\Phi}_{\text{tr}_e}\mathbf{w}||_2^2 + ||\mathbf{y}_{\text{p}} - \mathbf{\Phi}_{\text{p}_e}\mathbf{w}||_2^2 + \lambda ||\mathbf{w}_{1:(d-1)}||_2^2 \right), \tag{7.5}$$

where $\lambda = 0$ for non-regularised models and $\mathbf{w}_{1:(d-1)}$ denotes all the entries of $\mathbf{w}$ but the bias term. The constant $\frac{1}{2(n_{\text{tr}}+n_{\text{p}})}$ is typically useful for optimisation algorithms such as gradient descent to help in convergence, namely to adapt the learning rate to the size of the training set or mini-batch. This is because the norm of the gradient typically grows with the size of the training set. Nevertheless, this constant does not change the theoretical value of the optimal solution, so for our analysis we can leave it out. The well-known solution to the least-squares problem (7.5) is then:

$$\mathbf{w}^\star = \left(\mathbf{\Phi}_{\text{tr}_e}^\mathsf{T} \mathbf{\Phi}_{\text{tr}_e} + \mathbf{\Phi}_{\text{p}_e}^\mathsf{T} \mathbf{\Phi}_{\text{p}_e} + \lambda \mathbf{I}_{d_e} \right)^{-1} \left(\mathbf{\Phi}_{\text{tr}_e}^\mathsf{T} \mathbf{y}_{\text{tr}} + \mathbf{\Phi}_{\text{p}_e}^\mathsf{T} \mathbf{y}_{\text{p}} \right), \tag{7.6}$$

where it is clear that the dependency of the model's parameters on n_{p} is through $\mathbf{\Phi}_{\text{p}_e}$ and $\mathbf{y}_{\text{p}}$. In the previous equation, we define the block matrix $\mathbf{I}_{d_e} = \mathbf{I}_d - \mathbf{e}_1\mathbf{e}_1^\mathsf{T} = \begin{bmatrix} 0 & \mathbf{0}_{d-1}^\mathsf{T} \\ \mathbf{0}_{d-1} & \mathbf{I}_{d-1} \end{bmatrix}$, where $\mathbf{I}_{d-1} \in \mathbb{R}^{(d-1)\times(d-1)}$ is the identity matrix, $\mathbf{0}_{d-1} \in \mathbb{R}^{d-1}$ is a column vector of zeros, and $\mathbf{e}_1 = [1\ \mathbf{0}_{d-1}^\mathsf{T}]^\mathsf{T}$ is a basis vector.

In the following derivations, we shall assume that the scatter matrix of the clean training data, $\mathbf{S}_{\text{tr}} = \mathbf{\Phi}_{\text{tr}_e}^\mathsf{T} \mathbf{\Phi}_{\text{tr}_e}$, and its regularised version, $\mathbf{S}_{\text{tr}_\lambda} = \mathbf{\Phi}_{\text{tr}_e}^\mathsf{T} \mathbf{\Phi}_{\text{tr}_e} + \lambda \mathbf{I}_{d_e}$, are positive definite (and hence non-singular),[4] and that the poisoning points follow a distribution different from the clean distribution. We shall also assume that n_{p} and n_{tr} are of comparable magnitude, i.e., one of them is not much larger than the other. Moreover, we define the poisoning scatter matrix $\mathbf{S}_{\text{p}} = \mathbf{\Phi}_{\text{p}_e}^\mathsf{T} \mathbf{\Phi}_{\text{p}_e}$.

As shown in Appx. D, the scatter matrix of the complete training data: $\mathbf{S} = \mathbf{S}_{\text{tr}} + \mathbf{S}_{\text{p}}$ and its regularised version $\mathbf{S}_\lambda = \mathbf{S} + \lambda \mathbf{I}_{d_e}$ are positive definite, because $\mathbf{S}_{\text{tr}}$ and $\mathbf{S}_{\text{tr}_\lambda}$ are assumed to be positive definite.

[4] This is usually the case if the number of training points linearly independent in the basis functions space is equal to or greater than the number of basis functions (including the dummy basis function, i.e., the one corresponding to the bias term of the model).

7.4.1 Non-Regularised Linear Regression

Let us focus on a non-regularised poisoned linear regression model. We shall start with the steps #1 and #2 of the poisonability problem (Subsect. 7.3.2), in which we aim to find a mapping between the clean and poisoned model's parameters.

Obtaining a Mapping Between the Clean Parameters and the Poisoned Parameters

For linear regression, it can be shown that the poisoned parameters $\mathbf{w}^\star$ are an affine transformation of the clean parameters $\mathbf{w}^\star_{\mathrm{tr}} = \mathbf{S}_{\mathrm{tr}}^{-1}\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T}\mathbf{y}_{\mathrm{tr}}$:

$$\mathbf{w}^\star = \mathbf{A}_{w^\star \leftarrow w^\star_{\mathrm{tr}}}\mathbf{w}^\star_{\mathrm{tr}} + \mathbf{b}_{w^\star \leftarrow w^\star_{\mathrm{tr}}}, \tag{7.7}$$

where, by using the Woodbury formula (Appx. A.1), and assuming that the Schur complement of $\mathbf{S}_{\mathrm{tr}}$ (Appx. A.1):

$$\mathbf{S}_{\mathrm{tr}_S} = \mathbf{I}_{n_{\mathrm{p}}} + \mathbf{\Phi}_{\mathrm{p}_e}\mathbf{S}_{\mathrm{tr}}^{-1}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T} \tag{7.8}$$

is also non-singular, we obtain:

$$\mathbf{A}_{w^\star \leftarrow w^\star_{\mathrm{tr}}} = \mathbf{I}_d - \mathbf{S}_{\mathrm{tr}}^{-1}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{S}_{\mathrm{tr}_S}^{-1}\mathbf{\Phi}_{\mathrm{p}_e}, \tag{7.9}$$

$$\mathbf{b}_{w^\star \leftarrow w^\star_{\mathrm{tr}}} = \mathbf{S}_{\mathrm{tr}}^{-1}\mathbf{A}_{w^\star \leftarrow w^\star_{\mathrm{tr}}}^\mathsf{T}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_{\mathrm{p}}, \tag{7.10}$$

where $\mathbf{I}_d \in \mathbb{R}^{d\times d}$ and $\mathbf{I}_{n_{\mathrm{p}}} \in \mathbb{R}^{n_{\mathrm{p}}\times n_{\mathrm{p}}}$ are identity matrices.

Interestingly, in the particular case where the poisoning scatter matrix, $\mathbf{S}_{\mathrm{p}}$, is non-singular, i.e., if $\mathrm{rank}(\mathbf{\Phi}_{\mathrm{p}_e}) = d$ (see Appx. E.1), and that its Schur complement:

$$\mathbf{S}_{\mathrm{p}_S} = \mathbf{I}_{n_{\mathrm{p}}} + \mathbf{\Phi}_{\mathrm{tr}_e}\mathbf{S}_{\mathrm{p}}^{-1}\mathbf{\Phi}_{\mathrm{tr}_e}^\mathsf{T} \tag{7.11}$$

is non-singular, we can also decompose the poisoned parameters as:

$$\mathbf{w}^\star = \mathbf{A}_{w^\star \leftarrow w^\star_{\mathrm{tr}}}\mathbf{w}^\star_{\mathrm{tr}} + \mathbf{A}_{w^\star \leftarrow w^\star_{\mathrm{p}}}\mathbf{w}^\star_{\mathrm{p}}, \tag{7.12}$$

where

$$\mathbf{w}^\star_{\mathrm{p}} = \mathbf{S}_{\mathrm{p}}^{-1}\mathbf{\Phi}_{\mathrm{p}_e}^\mathsf{T}\mathbf{y}_{\mathrm{p}} \tag{7.13}$$

are the parameters learnt exclusively on the poisoning points and:

$$\mathbf{A}_{w^\star \leftarrow w^\star_{\mathrm{p}}} = \mathbf{I}_d - \mathbf{S}_{\mathrm{p}}^{-1} \mathbf{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}} \mathbf{S}_{\mathrm{p}S}^{-1} \mathbf{\Phi}_{\mathrm{tr}_e}. \tag{7.14}$$

In words, under the previous assumptions, Eq. (7.12) tells us that the poisoned parameters, $\mathbf{w}^\star$, can be decomposed as the sum of a linear transformation of $\mathbf{w}^\star_{\mathrm{tr}}$ and a linear transformation of $\mathbf{w}^\star_{\mathrm{p}}$. This shows how the clean and poisoning distributions combine together in the poisoned parameters.

Objective Function

After computing the mapping between the clean parameters and the poisoned parameters, we can construct the objective function $p(\hat{n}_{\mathrm{p}}, \xi_{\mathrm{p}})$[5] (step #3 of the poisonability problem, Subsect. 7.3.2):

$$p(\hat{n}_{\mathrm{p}}, \xi_{\mathrm{p}}) = \left(\hat{n}_{\mathrm{p}}^{-1} ||\mathbf{y}_{\mathrm{p}} - \mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star||_2^2 - \xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1} ||\mathbf{y}_{\mathrm{tr}} - \mathbf{\Phi}_{\mathrm{tr}_e} \mathbf{w}^\star||_2^2 \right) \mathbb{1}_{\hat{n}_{\mathrm{p}} \geq 1}(\hat{n}_{\mathrm{p}}), \tag{7.15}$$

where $\mathbb{1}_{\hat{n}_{\mathrm{p}} \geq 1}(\hat{n}_{\mathrm{p}})$ is the indicator function, defined as $\mathbb{1}_{\hat{n}_{\mathrm{p}} \geq 1}(\hat{n}_{\mathrm{p}}) = 1$ if $\hat{n}_{\mathrm{p}} \geq 1$, and $\mathbb{1}_{\hat{n}_{\mathrm{p}} \geq 1}(\hat{n}_{\mathrm{p}}) = 0$ otherwise. Once we have built the objective function, we can find its roots (steps #4 of the poisonability problem, Subsect. 7.3.2) from the following inequality:

$$\left(\hat{n}_{\mathrm{p}}^{-1} ||\mathbf{y}_{\mathrm{p}} - \mathbf{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star||_2^2 - \xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1} ||\mathbf{y}_{\mathrm{tr}} - \mathbf{\Phi}_{\mathrm{tr}_e} \mathbf{w}^\star||_2^2 \right) \mathbb{1}_{\hat{n}_{\mathrm{p}} \geq 1}(\hat{n}_{\mathrm{p}}) < 0. \tag{7.16}$$

In the general case, determining $\hat{n}_{\mathrm{p}}$ and the conditions for $\mathbf{\Phi}_{\mathrm{p}_e}$ and $\mathbf{y}_{\mathrm{p}}$ that satisfy the inequality (7.16) is not analytically tractable, because $\mathbf{w}^\star$, $\mathbf{\Phi}_{\mathrm{p}_e}$ and $\mathbf{y}_{\mathrm{p}}$ are functions of $\hat{n}_{\mathrm{p}}$. As a first approximation, let us assume that $\mathcal{D}_{\mathrm{p}}$ is a set of n_{p} identical points (that are different from $\mathcal{D}_{\mathrm{tr}}$), i.e., $\mathcal{D}_{\mathrm{p}} = \{(\mathbf{x}_{\mathrm{p}}, y_{\mathrm{p}})\}_{i=1}^{n_{\mathrm{p}}}$, which implies that $\mathbf{\Phi}_{\mathrm{p}_e}$ has rank-1.

This assumption is reasonable, for example, if the attacker has some particular target parameters in mind, $\mathbf{w}^\star_{\mathrm{target}}$, whose bias term is zero, i.e., $w^\star_{\mathrm{target}_0} = 0$. Then, in line with the teaching dimension of linear learners (Liu et al., 2016), the attack can be characterised by the single point $\left(\boldsymbol{\phi}_{\mathrm{p}}, y_{\mathrm{p}} = \boldsymbol{\phi}_{\mathrm{p}}^{\mathsf{T}} \mathbf{w}^\star_{\mathrm{target}} \right)$ or $\left(\boldsymbol{\phi}_{\mathrm{p}} = c\mathbf{w}^\star_{\mathrm{target}}, y_{\mathrm{p}} = c||\mathbf{w}^\star_{\mathrm{target}}||_2^2 \right)$, where $c \in \mathbb{R}$. Both tuples are instances of machine teaching (Liu et al., 2016).

In the case of a rank-1 attack, using the Sherman-Morrison formula (Appx. A.2, special

[5] Recall that the poisonability intolerance, ξ_{p}, is a given hyperparameter.

case of Woodbury formula), it is not difficult to show that:

$$\mathbf{A}_{w^\star \leftarrow w^\star_{\text{tr}}} = \mathbf{I}_d - \frac{\hat{n}_{\text{p}}}{1 + \hat{n}_{\text{p}} d_{\text{p}}} \mathbf{S}_{\text{tr}}^{-1} \boldsymbol{\phi}_{\text{p}_e} \boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}}, \tag{7.17}$$

$$\mathbf{b}_{w^\star \leftarrow w^\star_{\text{tr}}} = \frac{\hat{n}_{\text{p}}}{1 + \hat{n}_{\text{p}} d_{\text{p}}} \mathbf{S}_{\text{tr}}^{-1} \boldsymbol{\phi}_{\text{p}_e} y_{\text{p}}, \tag{7.18}$$

where $d_{\text{p}} = \boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}} \mathbf{S}_{\text{tr}}^{-1} \boldsymbol{\phi}_{\text{p}_e} > 0$ (since $\mathbf{S}_{\text{tr}}$ is positive definite) is the quadratic bilinear form, in particular the inner product, of the poisoning point over the inverse scatter matrix of the clean training data. This inner product is related to the squared Mahalanobis norm of the poisoning point over the clean training data. It can also be interpreted as applying the change of basis $\mathbf{S}_{\text{tr}}^{-1/2}$ (positive definite) to the poisoning point $\boldsymbol{\phi}_{\text{p}_e}$, and then taking the dot product of the result.

Moreover, given the definition of the poisoning points, the following equality holds true:

$$\hat{n}_{\text{p}}^{-1} ||\mathbf{y}_{\text{p}} - \boldsymbol{\Phi}_{\text{p}_e} \mathbf{w}^\star||_2^2 = \hat{n}_{\text{p}}^{-1} \sum_{i=1}^{\hat{n}_{\text{p}}} (y_{\text{p}} - \boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}} \mathbf{w}^\star)^2 = (y_{\text{p}} - \boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}} \mathbf{w}^\star)^2 \mathbb{1}_{\hat{n}_{\text{p}} \geq 1}(\hat{n}_{\text{p}}), \tag{7.19}$$

which leads to the objective function:

$$p(\hat{n}_{\text{p}}, \xi_{\text{p}}) = \left(\left(y_{\text{p}} - \boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}} \mathbf{w}^\star \right)^2 - \xi_{\text{p}} n_{\text{tr}}^{-1} \, ||\mathbf{y}_{\text{tr}} - \boldsymbol{\Phi}_{\text{tr}_e} \mathbf{w}^\star||_2^2 \right) \mathbb{1}_{\hat{n}_{\text{p}} \geq 1}(\hat{n}_{\text{p}}) < 0, \tag{7.20}$$

which is the feasible region of the problem. If we evaluate $p(\hat{n}_{\text{p}}, \xi_{\text{p}}) = 0$, after applying some algebraic operations in (7.20), we obtain:

$$\left(a_{2_{\hat{n}_{\text{p}}}} \hat{n}_{\text{p}}^2 + a_{1_{\hat{n}_{\text{p}}}} \hat{n}_{\text{p}} + a_{0_{\hat{n}_{\text{p}}}} \right) \mathbb{1}_{\hat{n}_{\text{p}} \geq 1}(\hat{n}_{\text{p}}) = 0, \tag{7.21}$$

with the following coefficients:

$$a_{2_{\hat{n}_{\text{p}}}} = -d_{\text{p}} (\hat{e}_{\text{p}}^2 + d_{\text{p}} n_{\text{tr}} \epsilon_{\text{tr}}^2) = d_{\text{p}} \left(\frac{a_{1_{\hat{n}_{\text{p}}}}}{2} - \hat{e}_{\text{p}}^2 \right) < 0, \tag{7.22}$$

$$a_{1_{\hat{n}_{\text{p}}}} = -2 d_{\text{p}} n_{\text{tr}} \epsilon_{\text{tr}}^2 < 0, \tag{7.23}$$

$$a_{0_{\hat{n}_{\text{p}}}} = n_{\text{tr}} (\xi_{\text{p}}^{-1} \hat{e}_{\text{p}}^2 - \epsilon_{\text{tr}}^2), \tag{7.24}$$

where $\epsilon_{\text{tr}} = n_{\text{tr}}^{-1/2} ||\mathbf{y}_{\text{tr}} - \boldsymbol{\Phi}_{\text{tr}_e} \mathbf{w}^\star_{\text{tr}}||_2 \geq 0$ is the root-mean-squared error (RMSE) of the clean training samples on the clean model, i.e., the mean of the clean residuals, which can be

interpreted as an approximation of the "natural"[6] additive noise; and $\hat{e}_\mathrm{p} = y_\mathrm{p} - \boldsymbol{\phi}_{\mathrm{p}_e}^\mathsf{T} \mathbf{w}_\mathrm{tr}^\star$ is the poisoning residual.

It is noteworthy that the polynomial $p(\hat{n}_\mathrm{p}, \xi_\mathrm{p})$ is concave in $\hat{n}_\mathrm{p}$, and thus its maximum is located at $-\frac{a_{1_{\hat{n}_\mathrm{p}}}}{2a_{2_{\hat{n}_\mathrm{p}}}}$. Since $-\frac{a_{1_{\hat{n}_\mathrm{p}}}}{2a_{2_{\hat{n}_\mathrm{p}}}} < 0$. This shows that $p(\hat{n}_\mathrm{p}, \xi_\mathrm{p})$ has at most one root in its feasible domain. There are two cases for the remaining root: (i) $\hat{n}_\mathrm{p}^\star \leq 1$ (which includes the case where the root is negative), or (ii) $\hat{n}_\mathrm{p}^\star > 1$. The first case means that the tolerable error of a single poisoning point is lower than the mean error of the clean points on the poisoned model, and therefore $n_{\mathrm{p}_\mathrm{inf}}^\star = n_{\mathrm{p}_\mathrm{sup}}^\star = 1$.

Corollary 7.1. *A non-regularised linear regression model poisoned with n_p identical poisoning points has a single breaking point, i.e., $|\mathcal{N}_\mathrm{break}^\star| = 1$, $n_{\mathrm{p}_\mathrm{inf}}^\star = n_{\mathrm{p}_\mathrm{sup}}^\star = n_\mathrm{p}^\star$.*

Solution to the Poisonability Problem and Interpretation

Given the second-degree polynomial (7.21), we can calculate its closed-form roots. Then, $\exists \hat{n}_\mathrm{p}^\star \mid \hat{n}_\mathrm{p}^{\star^{-1}} ||\mathbf{y}_\mathrm{p} - \boldsymbol{\Phi}_{\mathrm{p}_e} \mathbf{w}^\star||_2^2 = \xi_\mathrm{p} n_\mathrm{tr}^{-1} ||\mathbf{y}_\mathrm{tr} - \boldsymbol{\Phi}_{\mathrm{tr}_e} \mathbf{w}^\star||_2^2$ and it can be computed as:

$$\hat{n}_\mathrm{p}^\star = \frac{-d_\mathrm{p} n_\mathrm{tr} \epsilon_\mathrm{tr}^2 + |\hat{e}_\mathrm{p}| \sqrt{d_\mathrm{p} n_\mathrm{tr} (\hat{e}_\mathrm{p}^2 \xi_\mathrm{p}^{-1} + \epsilon_\mathrm{tr}^2 (d_\mathrm{p} \xi_\mathrm{p}^{-1} n_\mathrm{tr} - 1))}}{d_\mathrm{p} (d_\mathrm{p} n_\mathrm{tr} \epsilon_\mathrm{tr}^2 + \hat{e}_\mathrm{p}^2)}. \tag{7.25}$$

Eq. (7.25) can also be expressed in a more interpretable manner as:

$$\hat{n}_\mathrm{p}^\star \left(\xi_\mathrm{p} n_\mathrm{tr}^{-1}, 1 + \mathrm{ANR}, d_\mathrm{p} \right) = \frac{(1 + \mathrm{ANR}) \sqrt{1 + (\xi_\mathrm{p} n_\mathrm{tr}^{-1}) d_\mathrm{p}^{-1} ((1 + \mathrm{ANR})^2 - 1)} - 1}{(\xi_\mathrm{p} n_\mathrm{tr}^{-1})(1 + \mathrm{ANR})^2 + d_\mathrm{p}}, \tag{7.26}$$

where $\mathrm{ANR} = \frac{\epsilon_\mathrm{p}}{\epsilon_\mathrm{tr}}$ is the tolerable attack-to-noise ratio, and $\epsilon_\mathrm{p} = \xi_\mathrm{p}^{-1/2} |\hat{e}_\mathrm{p}| - \epsilon_\mathrm{tr}$ can be interpreted as an approximation of the additive poisoning noise tolerable in the model, according to the poisonability tolerance. Hence, we show that the point of the model depends on $\xi_\mathrm{p} n_\mathrm{tr}^{-1}$, $1 + \mathrm{ANR}$, and a surrogate of the squared Mahalanobis norm of the poisoning point, d_p.

Now, there are two possible scenarios: $\hat{n}_\mathrm{p}^\star$ has real or complex roots, which can be derived from (7.25):

$$\hat{n}_\mathrm{p}^\star \in \begin{cases} \mathbb{R} & \text{if } \xi_\mathrm{p} \leq \epsilon_\mathrm{tr}^{-2} \hat{e}_\mathrm{p}^2 + d_\mathrm{p} n_\mathrm{tr}, \\ \mathbb{C} & \text{otherwise.} \end{cases} \tag{7.27}$$

[6] Here we refer "natural" to the randomness of the measurement process. It represents the aleatory uncertainty captured in the observations. This could be for example sensor noise or motion noise.

Note that $\hat{n}_{\mathrm{p}}^{\star} \in \mathbb{C}$ implies that the truncated polynomial does not have roots in its domain. Hence, since $a_{2_{\hat{n}_{\mathrm{p}}}} < 0$, $p(\hat{n}_{\mathrm{p}})$ is concave, and $p(\hat{n}_{\mathrm{p}}) < 0 \; \forall \hat{n}_{\mathrm{p}} \in \mathbb{R}_{\geq 1}$, this means that the mean error of the clean points on the poisoned model is larger than the mean error of the poisoning points on the poisoned model. Because the poisonability intolerance is too large, the tolerable poisoning distribution dominates the clean distribution.

On the other hand, to recover the actual positive integer variable $n_{\mathrm{p}}^{\star}$, we can calculate it as:

$$n_{\mathrm{p}}^{\star} = \begin{cases} \left\lceil \hat{n}_{\mathrm{p}}^{\star} \right\rceil & \text{if } \hat{n}_{\mathrm{p}}^{\star} > 1, \\ 1 & \text{otherwise.} \end{cases} \tag{7.28}$$

Bounds on the Value of the Breaking Point

To gain intuition into the behaviour of the breaking point, $\hat{n}_{\mathrm{p}}^{\star}$, depending on the characteristics of the training set, we shall analyse some bounds.

In the extreme case where the clean training dataset is very noisy, i.e., $\epsilon_{\mathrm{tr}} \to \infty$ and therefore ANR $\to -1$. Alternatively, if $\hat{e}_{\mathrm{p}} \to 0$, ANR $\to -1$ as well, which can be the case for very targeted attacks, which typically require a lower number of poisoning points, because the poisoning points can be typically closer to the regression hyperplane. The number of poisoning points required $\hat{n}_{\mathrm{p}}^{\star}$ is:

$$\lim_{\mathrm{ANR} \to -1} \hat{n}_{\mathrm{p}}^{\star} \left(\xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1}, 1 + \mathrm{ANR}, d_{\mathrm{p}} \right) = -d_{\mathrm{p}}^{-1} \Rightarrow n_{\mathrm{p}}^{\star} = 1 \tag{7.29}$$

In this case, only one poisoning points is needed because the mean error of the clean points is already too large compared to the mean error of the poisoning points on the clean model. This can be interpreted such that the poisoning points are completely hidden in the natural noise.

In the other extreme case where the clean training dataset is not noisy, $\epsilon_{\mathrm{tr}} \to 0$ and therefore ANR $\to \infty$, we need to compute

$$\lim_{\mathrm{ANR} \to \infty} \hat{n}_{\mathrm{p}}^{\star} \left(\xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1}, 1 + \mathrm{ANR}, d_{\mathrm{p}} \right) = \sqrt{d_{\mathrm{p}}^{-1} \xi_{\mathrm{p}}^{-1} n_{\mathrm{tr}}}, \tag{7.30}$$

which provides an upper bound on the effort needed by the attacker.

Finally, when the number of clean training samples is very large, i.e., $n_{\text{tr}}^{-1} \to 0$:

$$\lim_{n_{\text{tr}}^{-1}\to 0} \hat{n}_{\text{p}}^{\star}\left(\xi_{\text{p}} n_{\text{tr}}^{-1}, 1 + \text{ANR}, d_{\text{p}}\right) = \lim_{n_{\text{tr}}^{-1}\to 0} d_{\text{p}}^{-1}\ \text{ANR}, \tag{7.31}$$

where d_{p}^{-1} and ANR are functions of n_{tr}. As for d_{p}^{-1}:

$$\begin{aligned}
\lim_{n_{\text{tr}}\to\infty} d_{\text{p}}(n_{\text{tr}}) &= \lim_{n_{\text{tr}}\to\infty} \boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}} \mathbf{S}_{\text{tr}}^{-1} \boldsymbol{\phi}_{\text{p}_e} = \lim_{n_{\text{tr}}\to\infty} ||\boldsymbol{\phi}_{\text{p}_e}^{\mathsf{T}} \mathbf{S}_{\text{tr}}^{-1} \boldsymbol{\phi}_{\text{p}_e}||_2 \\
&\leq \lim_{n_{\text{tr}}\to\infty} ||\boldsymbol{\phi}_{\text{p}_e}||_2^2 ||\mathbf{S}_{\text{tr}}^{-1}||_2 = ||\boldsymbol{\phi}_{\text{p}_e}||_2^2 \lim_{n_{\text{tr}}\to\infty} \max_i \sigma_i(\mathbf{S}_{\text{tr}}^{-1}) \\
&= ||\boldsymbol{\phi}_{\text{p}_e}||_2^2 \lim_{n_{\text{tr}}\to\infty} \max_i \lambda_i(\mathbf{S}_{\text{tr}}^{-1}) = ||\boldsymbol{\phi}_{\text{p}_e}||_2^2 \lim_{n_{\text{tr}}\to\infty} \max_i \lambda_i(\mathbf{S}_{\text{tr}})^{-1} \\
&= ||\boldsymbol{\phi}_{\text{p}_e}||_2^2 \max_i \lim_{n_{\text{tr}}\to\infty} \lambda_i \left(\sum_{j=1}^{n_{\text{tr}'}} \boldsymbol{\phi}_{\text{tr}_{e_j}} \boldsymbol{\phi}_{\text{tr}_{e_j}}^{\mathsf{T}} + \sum_{j'=n'_{\text{tr}}+1}^{n_{\text{tr}}} \boldsymbol{\phi}_{\text{tr}_{e_{j'}}} \boldsymbol{\phi}_{\text{tr}_{e_{j'}}}^{\mathsf{T}} \right)^{-1} \\
&= ||\boldsymbol{\phi}_{\text{p}_e}||_2^2 \max_i \lim_{n_{\text{tr}}\to\infty} \left(\lambda_i \left(\sum_{j=1}^{n_{\text{tr}'}} \boldsymbol{\phi}_{\text{tr}_{e_j}} \boldsymbol{\phi}_{\text{tr}_{e_j}}^{\mathsf{T}} \right) + \sum_{j'=n'_{\text{tr}}+1}^{n_{\text{tr}}} \nu_{j'} \right)^{-1} = 0,
\end{aligned} \tag{7.32}$$

where $0 < n'_{\text{tr}} < n_{\text{tr}}$; $\left|\left|\mathbf{S}_{\text{tr}}^{-1}\right|\right|_2 = \max_i \sigma_i\left(\mathbf{S}_{\text{tr}}^{-1}\right)$, $i = 1, \ldots, d$, represents the spectral norm of $\mathbf{S}_{\text{tr}}^{-1}$; $\sigma_i\left(\mathbf{S}_{\text{tr}}^{-1}\right) = \lambda_i\left(\mathbf{S}_{\text{tr}}^{-1}\right) > 0\ \forall i$ (because $\mathbf{S}_{\text{tr}}^{-1}$ is positive definite) denote the ith singular value and the ith eigenvalue of $\mathbf{S}_{\text{tr}}^{-1}$, respectively; the inequality is a direct consequence of the definition of an induced norm of a matrix; and $\nu_{j'} > 0\ \forall j'$ as a result of $\boldsymbol{\phi}_{\text{tr}_{e_{j'}}} \boldsymbol{\phi}_{\text{tr}_{e_{j'}}}^{\mathsf{T}}$ being positive definite $\forall j'$ and Weyl's theorem on eigenvalues (Weyl, 1912). Therefore, $\lim_{n_{\text{tr}}^{-1}\to 0} d_{\text{p}}^{-1} = \infty$.

Now, if we assume that the poisoning points are constrained in such a way that the tolerable poisoning noise, ϵ_{p}, is negative, according to the following definition:

$$\lim_{n_{\text{tr}}\to\infty} \epsilon_{\text{p}}(n_{\text{tr}}) = \lim_{n_{\text{tr}}\to\infty} \xi_{\text{p}}^{-1/2} |\hat{e}_{\text{p}}| - \epsilon_{\text{tr}} < 0, \tag{7.33}$$

then ANR < 0 and

$$\lim_{n_{\text{tr}}^{-1}\to 0, \text{ANR}<0} \hat{n}_{\text{p}}^{\star}\left(\xi_{\text{p}} n_{\text{tr}}^{-1}, 1 + \text{ANR}, d_{\text{p}}\right) = -\infty\ \Rightarrow n_{\text{p}}^{\star} = 1, \tag{7.34}$$

which shows that only one poisoning point is needed in this case, because its corresponding error is very low. On the contrary, if we assume that the poisoning points are less

constrained in such a way that ϵ_{p} is positive:

$$\lim_{n_{\mathrm{tr}}\to\infty} \epsilon_{\mathrm{p}}(n_{\mathrm{tr}}) = \lim_{n_{\mathrm{tr}}\to\infty} \xi_{\mathrm{p}}^{-1/2}|\hat{e}_{\mathrm{p}}| - \epsilon_{\mathrm{tr}} > 0, \tag{7.35}$$

then ANR > 0 and

$$\lim_{n_{\mathrm{tr}}^{-1}\to 0,\mathrm{ANR}>0} \hat{n}_{\mathrm{p}}^{\star}\left(\xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1}, 1+\mathrm{ANR}, d_{\mathrm{p}}\right) = \infty, \tag{7.36}$$

which shows that, if the poisoning points are less constrained, augmenting the training set with a large number of clean training points prevents the attacker's success.

As a numerical example, let us assume a poisonability intolerance $\xi_{\mathrm{p}} = 1$, a dataset of $n_{\mathrm{tr}} = 1,000$ clean points, $\epsilon_{\mathrm{tr}} = 0.1$, and an attacker's budget of $d_{\mathrm{p}} = 2n_{\mathrm{tr}}^{-1}$ and $|\hat{e}_{\mathrm{p}}| = \frac{3}{2}\epsilon_{\mathrm{tr}}$. In this case, $\mathrm{ANR} = \frac{\epsilon_{\mathrm{p}}}{\epsilon_{\mathrm{tr}}} = \frac{1}{2}$ and $n_{\mathrm{p}}^{\star} = 215$ poisoning points, i.e., 17.70% of the training set. However, if we set a larger poisonability intolerance, e.g., $\xi_{\mathrm{p}} = 2$, $\mathrm{ANR} \approx 60.66 \cdot 10^{-3}$ and $n_{\mathrm{p}}^{\star} = 30$ poisoning points, i.e., only 2.91% of the training set.

So far, we have analysed the poisonability of non-regularised linear regression. Under a rank-1 attack, we have analytically determined the value of the breaking point of the model, i.e., the number of poisoning points needed so that the mean error of poisoning points matches the tolerable mean error of the clean points on the poisoned model. We have also analysed some bounds on the breaking and obtained that, for instance, if the poisoning points are not constrained, augmenting the training set with a large number of clean points increases the value of the breaking point, making the model more robust. Next, we will study the poisonability of ridge regression, whose solutions can also be found analytically.

7.4.2 Ridge Regression

Let us now analyse a ridge regression model—L_2 regularisation—with poisoned parameters $\mathbf{w}_{\lambda}^{\star} \in \mathbb{R}^d$, and regularisation hyperparameter $\lambda > 0$. As in Subsect. 6.2.1, we set $\mathcal{D}_{\mathrm{p}}$ as a set of n_{p} identical points (that are different from $\mathcal{D}_{\mathrm{tr}}$), i.e., $\mathcal{D}_{\mathrm{p}} = \{(\mathbf{x}_{\mathrm{p}}, y_{\mathrm{p}})\}_{i=1}^{n_{\mathrm{p}}}$. In other words, we study a rank-1 attack.

This assumption is reasonable, for example, if the attacker has some particular target parameters in mind, $\mathbf{w}_{\mathrm{target}}^{\star}$, whose bias term is zero, i.e., $w_{\mathrm{target}_0}^{\star} = 0$, and the attacker is aware of the defender's regularisation hyperparameter. Then, according to the teaching

dimension of linear learners (Liu et al., 2016), the attack can be characterised by the single point ($\boldsymbol{\phi}_{\mathrm{p}} = c\mathbf{w}^{\star}_{\mathrm{target}}$, $y_{\mathrm{p}} = c||\mathbf{w}^{\star}_{\mathrm{target}}||_2^2 + c^{-1}\lambda$), where $c \in \mathbb{R}$. In this example, the poisoning label intentionally overshoots $c||\mathbf{w}^{\star}_{\mathrm{target}}||_2^2$ in order to precisely counter the defender's regulariser.

Now, the poisoned parameters $\mathbf{w}^{\star}_{\lambda}$ are computed from the clean parameters $\mathbf{w}^{\star}_{\mathrm{tr}_\lambda}$ as:

$$\mathbf{w}^{\star}_{\lambda} = \mathbf{A}_{w^{\star}_{\lambda} \leftarrow w^{\star}_{\mathrm{tr}_\lambda}} \mathbf{w}^{\star}_{\mathrm{tr}_\lambda} + \mathbf{b}_{w^{\star}_{\lambda} \leftarrow w^{\star}_{\mathrm{tr}_\lambda}}, \tag{7.37}$$

where the clean parameters $\mathbf{w}^{\star}_{\mathrm{tr}_\lambda} = \mathbf{S}^{-1}_{\mathrm{tr}_\lambda} \boldsymbol{\Phi}^{\mathsf{T}}_{\mathrm{tr}_e} \mathbf{y}_{\mathrm{tr}}$. Under our assumptions,

$$\mathbf{A}_{w^{\star}_{\lambda} \leftarrow w^{\star}_{\mathrm{tr}_\lambda}} = \mathbf{I}_d - \frac{\hat{n}_{\mathrm{p}_\lambda}}{1 + \hat{n}_{\mathrm{p}_\lambda} d_{\mathrm{p}_\lambda}} \mathbf{S}^{-1}_{\mathrm{tr}_\lambda} \boldsymbol{\phi}_{\mathrm{p}_e} \boldsymbol{\phi}^{\mathsf{T}}_{\mathrm{p}_e}, \tag{7.38}$$

$$\mathbf{b}_{w^{\star}_{\lambda} \leftarrow w^{\star}_{\mathrm{tr}_\lambda}} = \frac{\hat{n}_{\mathrm{p}_\lambda}}{1 + \hat{n}_{\mathrm{p}_\lambda} d_{\mathrm{p}_\lambda}} \mathbf{S}^{-1}_{\mathrm{tr}_\lambda} \boldsymbol{\phi}_{\mathrm{p}_e} y_{\mathrm{p}}, \tag{7.39}$$

where $d_{\mathrm{p}_\lambda} = \boldsymbol{\phi}^{\mathsf{T}}_{\mathrm{p}_e} \mathbf{S}^{-1}_{\mathrm{tr}_\lambda} \boldsymbol{\phi}_{\mathrm{p}_e} > 0$ (since $\mathbf{S}_{\mathrm{tr}_\lambda}$ is positive definite) is the inner product of the poisoning point over the inverse regularised scatter matrix of the clean training data.

Objective Function and Solutions

Given the mapping (7.37), we can write the following objective function:

$$p(\hat{n}_{\mathrm{p}_\lambda}, \xi_{\mathrm{p}}) = \left(\left(y_{\mathrm{p}} - \boldsymbol{\phi}^{\mathsf{T}}_{\mathrm{p}_e} \mathbf{w}^{\star}_{\lambda} \right)^2 - \xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1} ||\mathbf{y}_{\mathrm{tr}} - \boldsymbol{\Phi}_{\mathrm{tr}_e} \mathbf{w}^{\star}_{\lambda}||_2^2 \right) \mathbb{1}_{\hat{n}_{\mathrm{p}_\lambda} \geq 1}(\hat{n}_{\mathrm{p}_\lambda}) < 0, \tag{7.40}$$

which is the feasible region of the problem. If we evaluate $p(\hat{n}_{\mathrm{p}}, \xi_{\mathrm{p}}) = 0$, after applying some algebraic operations in (7.40), we obtain:

$$\left(a_{2_{\hat{n}_{\mathrm{p}_\lambda}}} \hat{n}^2_{\mathrm{p}_\lambda} + a_{1_{\hat{n}_{\mathrm{p}_\lambda}}} \hat{n}_{\mathrm{p}_\lambda} + a_{0_{\hat{n}_{\mathrm{p}_\lambda}}} \right) \mathbb{1}_{\hat{n}_{\mathrm{p}_\lambda} \geq 1}(\hat{n}_{\mathrm{p}_\lambda}) = 0, \tag{7.41}$$

with the following coefficients:

$$a_{2_{\hat{n}_{\mathrm{p}_\lambda}}} = d_{\mathrm{p}_\lambda}(2\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda - d_{\mathrm{p}_\lambda} n_{\mathrm{tr}} \epsilon^2_{\mathrm{tr}_\lambda}) - \hat{e}^2_{\mathrm{p}_\lambda}\gamma_\lambda = d_{\mathrm{p}_\lambda}\left(\frac{a_{1_{\hat{n}_{\mathrm{p}_\lambda}}}}{2} + \hat{e}_{\mathrm{p}_\lambda}\delta_\lambda \right) - \hat{e}^2_{\mathrm{p}_\lambda}\gamma_\lambda < 0, \tag{7.42}$$

$$a_{1_{\hat{n}_{\mathrm{p}_\lambda}}} = 2(\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda - d_{\mathrm{p}_\lambda} n_{\mathrm{tr}} \epsilon^2_{\mathrm{tr}_\lambda}), \tag{7.43}$$

$$a_{0_{\hat{n}_{\mathrm{p}_\lambda}}} = n_{\mathrm{tr}}(\xi_{\mathrm{p}}^{-1} \hat{e}^2_{\mathrm{p}_\lambda} - \epsilon^2_{\mathrm{tr}_\lambda}), \tag{7.44}$$

where $\epsilon_{\text{tr}_\lambda} = n_{\text{tr}}^{-1/2}||\mathbf{y}_{\text{tr}} - \mathbf{\Phi}_{\text{tr}_e}\mathbf{w}^\star_{\text{tr}_\lambda}||_2 \geq 0$ is the RMSE of the clean training samples on the clean model, i.e., the mean of the clean residuals; $\hat{e}_{\text{p}_\lambda} = y_{\text{p}} - \boldsymbol{\phi}^\mathsf{T}_{\text{p}_e}\mathbf{w}^\star_{\text{tr}_\lambda}$ is the poisoning residual; $\gamma_\lambda = ||\mathbf{\Phi}_{\text{tr}_e}\mathbf{S}^{-1}_{\text{tr}_\lambda}\phi_{\text{p}_e}||_2^2 = \boldsymbol{\phi}^\mathsf{T}_{\text{p}_e}(\mathbf{S}_{\text{tr}} + 2\lambda\mathbf{I}_{d_e} + \lambda^2\mathbf{I}_{d_e}\mathbf{S}^{-1}_{\text{tr}}\mathbf{I}_{d_e})^{-1}\boldsymbol{\phi}_{\text{p}_e} > 0$; and $\delta_\lambda = (\mathbf{\Phi}_{\text{tr}_e}\mathbf{S}^{-1}_{\text{tr}_\lambda}\phi_{\text{p}_e})^\mathsf{T}(\mathbf{\Phi}_{\text{tr}_e}\mathbf{w}^\star_{\text{tr}_\lambda} - \mathbf{y}_{\text{tr}}) = \boldsymbol{\phi}^\mathsf{T}_{\text{p}_e}((\mathbf{I}_d + \lambda\mathbf{I}_{d_e}\mathbf{S}^{-1}_{\text{tr}})^{-1} - \mathbf{I}_d)\mathbf{w}^\star_{\text{tr}_\lambda}$. γ_λ is the inner product of the poisoning point over the space defined by the matrix $(\mathbf{S}_{\text{tr}} + 2\lambda\mathbf{I}_{d_e} + \lambda^2\mathbf{I}_{d_e}\mathbf{S}^{-1}_{\text{tr}}\mathbf{I}_{d_e})^{-1}$, and a value of $|\delta_\lambda|$ close to zero means that λ is either very low or very large. Notice that if $\lambda = 0$, then $\gamma_\lambda = d_{\text{p}}$ and $\delta_\lambda = 0$. Note also that the coefficients $a_{2_{\hat{n}_{\text{p}_\lambda}}}$, $a_{1_{\hat{n}_{\text{p}_\lambda}}}$ and $a_{0_{\hat{n}_{\text{p}_\lambda}}}$ are a generalisation of $a_{2_{\hat{n}_{\text{p}}}}$, $a_{1_{\hat{n}_{\text{p}}}}$ and $a_{0_{\hat{n}_{\text{p}}}}$ when $\lambda > 0$, which allows more flexibility to define the polynomial $p(\hat{n}_{\text{p}_\lambda}, \xi_{\text{p}})$. By inspecting these coefficients, it is clear that varying λ can change these coefficients and thus $\hat{n}^\star_{\text{p}_\lambda}$.

We shall show that $a_{2_{\hat{n}_{\text{p}_\lambda}}}$ is always negative, and hence the polynomial $p(\hat{n}_{\text{p}_\lambda}, \xi_{\text{p}})$ (7.42) is concave in $\hat{n}_{\text{p}_\lambda}$. First, by inspection, $a_{2_{\hat{n}_{\text{p}_\lambda}}}$ is a concave quadratic function in $\hat{e}_{\text{p}_\lambda}$, whose roots are:

$$\hat{e}^\star_{\text{p}_{\lambda_1}}, \hat{e}^\star_{\text{p}_{\lambda_2}} = d_{\text{p}_\lambda}\gamma_\lambda^{-1}\left(\delta_\lambda \pm \sqrt{\delta_\lambda^2 - \gamma_\lambda n_{\text{tr}}\epsilon^2_{\text{tr}_\lambda}}\right) \tag{7.45}$$

Since the coefficient $a_{2_{\hat{n}_{\text{p}_\lambda}}}$ is concave in $\hat{e}_{\text{p}_\lambda}$, it is always negative if and only if its roots are in the complex domain. In other words, $a_{2_{\hat{n}_{\text{p}_\lambda}}}$ is negative if and only if $\delta_\lambda^2 < \gamma_\lambda n_{\text{tr}}\epsilon^2_{\text{tr}_\lambda}$. Expanding the inequality:

$$\left((\mathbf{\Phi}_{\text{tr}_e}\mathbf{S}^{-1}_{\text{tr}_\lambda}\phi_{\text{p}_e})^\mathsf{T}(\mathbf{\Phi}_{\text{tr}_e}\mathbf{w}^\star_{\text{tr}_\lambda} - \mathbf{y}_{\text{tr}})\right)^2 < ||\mathbf{\Phi}_{\text{tr}_e}\mathbf{S}^{-1}_{\text{tr}_\lambda}\phi_{\text{p}_e}||_2^2||\mathbf{y}_{\text{tr}} - \mathbf{\Phi}_{\text{tr}_e}\mathbf{w}^\star_{\text{tr}_\lambda}||_2^2, \tag{7.46}$$

which holds true due to the Cauchy-Schwarz inequality. Thus, $a_{2_{\hat{n}_{\text{p}_\lambda}}}$ is always negative.

If $p(\hat{n}_{\text{p}_\lambda})$ has real roots, this implies that there is a range in $\hat{n}_{\text{p}_\lambda}$ in which the clean distribution dominates the tolerable poisoning distribution and, outside that range, the tolerable poisoning distribution dominates the clean distribution. If $p(\hat{n}_{\text{p}_\lambda})$ has complex roots, it is always negative and thus the tolerable poisoning distribution always dominates the clean distribution.

On the other hand, since the maximum of $p(\hat{n}_{\text{p}_\lambda}\xi_{\text{p}})$, located at $-\frac{a_{1_{\hat{n}_{\text{p}_\lambda}}}}{2a_{2_{\hat{n}_{\text{p}_\lambda}}}}$, is not necessarily negative, $p(\hat{n}_{\text{p}_\lambda}, \xi_{\text{p}})$ can have up to two roots in its feasible domain. Applying Sturm's theorem (Sturm, 1829) in the interval $\hat{n}_{\text{p}_\lambda} \in (1, \infty)$, we obtain that this happens only when:

$$a_{1_{\hat{n}_{\text{p}_\lambda}}} + 2a_{2_{\hat{n}_{\text{p}_\lambda}}} > 0 \quad \text{and} \quad \frac{a^2_{1_{\hat{n}_{\text{p}_\lambda}}}}{4a_{2_{\hat{n}_{\text{p}_\lambda}}}} < a_{0_{\hat{n}_{\text{p}_\lambda}}} < -\left(a_{1_{\hat{n}_{\text{p}_\lambda}}} + a_{2_{\hat{n}_{\text{p}_\lambda}}}\right). \tag{7.47}$$

That is, $p(\hat{n}_{\mathrm{p}_\lambda}, \xi_\mathrm{p})$ can have up to two roots in its feasible domain in the particular case[7] where the following three conditions are satisfied:

$$\gamma_\lambda < \hat{e}_{\mathrm{p}_\lambda}^{-2}\left(\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda(2d_{\mathrm{p}_\lambda}+1) - d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2(d_{\mathrm{p}_\lambda}+1)\right) \tag{7.48}$$

and

$$\xi_\mathrm{p}^{-1} > \hat{e}_{\mathrm{p}_\lambda}^{-2}\left(n_\mathrm{tr}^{-1}\frac{(\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda - d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2)^2}{d_{\mathrm{p}_\lambda}(2\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda - d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2) - \hat{e}_{\mathrm{p}_\lambda}^2\gamma_\lambda} + \epsilon_{\mathrm{tr}_\lambda}^2\right) \tag{7.49}$$

and

$$\xi_\mathrm{p}^{-1} < \hat{e}_{\mathrm{p}_\lambda}^{-2}\left(n_\mathrm{tr}^{-1}\left(d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2(d_{\mathrm{p}_\lambda}+2) + \hat{e}_{\mathrm{p}_\lambda}(\hat{e}_{\mathrm{p}_\lambda}\gamma_\lambda - 2\delta_\lambda(d_{\mathrm{p}_\lambda}+1))\right) + \epsilon_{\mathrm{tr}_\lambda}^2\right). \tag{7.50}$$

Corollary 7.2. *A ridge regression model poisoned with n_p identical poisoning points can have up to two breaking points, i.e., $|\mathcal{N}_\mathrm{break}^\star| \in \{1,2\}$.*

As in the case of non-regularised linear regression, we can compute the closed-form roots of the objective function (7.41). Then, $\exists \hat{n}_{\mathrm{p}_{\lambda_i}}^\star > 0 \mid \hat{n}_{\mathrm{p}_{\lambda_i}}^{\star^{-1}}||\mathbf{y}_\mathrm{p} - \mathbf{\Phi}_{\mathrm{p}_e}\mathbf{w}_\lambda^\star||_2^2 = \xi_\mathrm{p}n_\mathrm{tr}^{-1}||\mathbf{y}_\mathrm{tr} - \mathbf{\Phi}_{\mathrm{tr}_e}\mathbf{w}_\lambda^\star||_2^2,\ i=1,2$, and they can be computed as:

$$\hat{n}_{\mathrm{p}_{\lambda_1}}^\star, \hat{n}_{\mathrm{p}_{\lambda_2}}^\star = \frac{-d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2 + \hat{e}_{\mathrm{p}_\lambda}\delta_\lambda \pm |\hat{e}_{\mathrm{p}_\lambda}|\sqrt{\gamma_\lambda n_\mathrm{tr}(\xi_\mathrm{p}^{-1}\hat{e}_{\mathrm{p}_\lambda}^2 - \epsilon_{\mathrm{tr}_\lambda}^2) + \xi_\mathrm{p}^{-1}n_\mathrm{tr}d_{\mathrm{p}_\lambda}(d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2 - 2\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda) + \delta_\lambda^2}}{\hat{e}_{\mathrm{p}_\lambda}^2\gamma_\lambda + d_{\mathrm{p}_\lambda}(d_{\mathrm{p}_\lambda}n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2 - 2\hat{e}_{\mathrm{p}_\lambda}\delta_\lambda)}. \tag{7.51}$$

Again, there are two possible scenarios: $\hat{n}_{\mathrm{p}_{\lambda_i}}^\star$ has real or complex roots, which can be derived from (7.51):

$$\hat{n}_{\mathrm{p}_{\lambda_i}}^\star \in \begin{cases} \mathbb{R} & \text{if } \xi_\mathrm{p} \leq \frac{n_\mathrm{tr}(\hat{e}_{\mathrm{p}_\lambda}(\hat{e}_{\mathrm{p}_\lambda}\gamma_\lambda - 2d_{\mathrm{p}_\lambda}\delta_\lambda) + d_{\mathrm{p}_\lambda}^2 n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2)}{\gamma_\lambda n_\mathrm{tr}\epsilon_{\mathrm{tr}_\lambda}^2 - \delta_\lambda^2}, \\ \mathbb{C} & \text{otherwise.} \end{cases} \tag{7.52}$$

As in the case of non-regularised linear regression, note that $\hat{n}_{\mathrm{p}_{\lambda_i}}^\star \in \mathbb{C}$ implies that the mean error of the clean points is larger than the mean error of the poisoning points on the poisoned model for any number of poisoning points, because the poisonability intolerance is very large.

In summary, we have analysed the poisonability of both non-regularised linear regression

[7] When $\lambda = 0$, there is only up to one root in the feasible domain, as seen in Subsect. 7.4.1. Moreover, given the correlation between $a_{1_{\hat{n}_{\mathrm{p}_\lambda}}}$ and $a_{2_{\hat{n}_{\mathrm{p}_\lambda}}}$ (see Eq. (7.42)), and that the polynomial on the right-hand side of (7.48) is concave in d_{p_λ}, we hypothesise that the inequality (7.48) may be only satisfied for extreme values of $\boldsymbol{\phi}_\mathrm{p}$, y_p, and/or λ. However, we do not provide a proof in this case.

and ridge regression. Under a rank-1 attack, we have analytically determined the value of the breaking point of the models, that is, the number of poisoning points needed so that the mean error of poisoning points matches the tolerable mean error of the clean points on the poisoned model. The coefficients of the objective function for ridge regression (7.21) are a generalisation of the ones for non-regularised linear regression (7.41). By inspecting these coefficients, it is clear that varying λ can change the value of the coefficients and thus $\hat{n}^{\star}_{\mathrm{p}_\lambda}$. For instance, searching for the correct value of λ can help increase the value of the breaking point of the model. Next, we will discuss some possible extensions to our analysis.

7.4.3 Discussion on Possible Extensions

The previous derivation can be extended to other regularised least-squares problems. For example, assuming that the defender is not aware of the attacker's distribution, the mean posterior predictive distribution of Bayesian linear regression (in which λ would correspond to the precision (inverse variance) of the model's parameters and the variance of the noise $\neq 1$).

As before, assuming $\mathcal{D}_{\mathrm{p}}$ is a set of n_{p} identical points, the objective function for non-parametric models is:

$$p(\hat{n}_{\mathrm{p}_\lambda}, \xi_{\mathrm{p}}) = \left(\left(y_{\mathrm{p}} - y_{*_{\mathrm{p}}}\right)^2 - \xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1} \sum_{i=1}^{n_{\mathrm{tr}}} (y_{\mathrm{tr}_i} - y_{*_{\mathrm{tr}_i}})^2 \right) \mathbb{1}_{\hat{n}_{\mathrm{p}_\lambda} \geq 1}(\hat{n}_{\mathrm{p}_\lambda}) < 0, \tag{7.53}$$

where $y_{*_{\mathrm{p}}}$ and $y_{*_{\mathrm{tr}_i}}$ denote the prediction for a poisoning and a clean training point, correspondingly.

Kernel Methods

In the case of kernel methods that admit a closed-form solution, such as kernel ridge regression, Gaussian process regression (GPR),[8] or other generalisations of L_2-regularised regression in the reproducing kernel Hilbert space (RKHS),[9] we can express the mean of

[8] A Gaussian process (GP) is a collection of random variables, any number of which have a joint Gaussian distribution (Rasmussen and Williams, 2006). A GP is completely specified by its mean function and covariance function. Usually, for notational simplicity, the mean function of the GP is assumed to be zero. This can be achieved by subtracting the sample mean from the data, and then, assuming a zero-mean model. This way, the covariance function of the model is $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}'))$, where $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is a real stochastic process.

[9] Roughly speaking, an RKHS is a Hilbert space of functions which generalises the concept of vector representations to the space of functions, in which point evaluation is a continuous linear functional.

the posterior predictive distribution as:

$$y_* = \mathbf{k}_*^\mathsf{T}(\mathbf{K} + \lambda \mathbf{I}_{n_\mathrm{tr}+n_\mathrm{p}})^{-1}[\mathbf{y}_\mathrm{tr}\ \mathbf{y}_\mathrm{p}]^\mathsf{T}, \tag{7.54}$$

where the kernel, a.k.a. Gram matrix or covariance function, is:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\mathsf{T} = \begin{bmatrix} \mathbf{\Phi}_\mathrm{tr}\mathbf{\Phi}_\mathrm{tr}^\mathsf{T} & \mathbf{\Phi}_\mathrm{tr}\mathbf{\Phi}_\mathrm{p}^\mathsf{T} \\ \mathbf{\Phi}_\mathrm{p}\mathbf{\Phi}_\mathrm{tr}^\mathsf{T} & \mathbf{\Phi}_\mathrm{p}\mathbf{\Phi}_\mathrm{p}^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_\mathrm{tr} & \mathbf{K}_\mathrm{tr,p} \\ \mathbf{K}_\mathrm{tr,p}^\mathsf{T} & \mathbf{K}_\mathrm{p} \end{bmatrix}, \tag{7.55}$$

and $\mathbf{\Phi}$, $\mathbf{\Phi}_\mathrm{tr}$, and $\mathbf{\Phi}_\mathrm{p}$ are possibly infinite-dimensional. For instance, the squared exponential covariance function $k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2\ell^2}||\mathbf{x} - \mathbf{x}'||_2^2)$, with hyperparameter ℓ defining the characteristic length-scale, corresponds to a Bayesian linear regression model with an infinite number of basis functions (Rasmussen and Williams, 2006). Indeed, for every positive definite covariance function $k(\mathbf{x}, \mathbf{x}')$, there exists a (possibly infinite) expansion in terms of basis functions (Mercer's theorem (Mercer, 1909)). Moreover, $[\mathbf{K}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$; $[\mathbf{K}_\mathrm{tr}]_{i,j} = k(\mathbf{x}_{\mathrm{tr}_i}, \mathbf{x}_{\mathrm{tr}_j})$; $[\mathbf{K}_\mathrm{p}]_{i,j} = k(\mathbf{x}_\mathrm{p}, \mathbf{x}_\mathrm{p})$; $[\mathbf{K}_\mathrm{tr,p}]_{i,j} = k(\mathbf{x}_{\mathrm{tr}_i}, \mathbf{x}_\mathrm{p})$; and

$$\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_{\mathrm{tr}_1})\ \ldots\ k(\mathbf{x}_*, \mathbf{x}_{\mathrm{tr}_{n_\mathrm{tr}}})\ k(\mathbf{x}_*, \mathbf{x}_\mathrm{p})\mathbf{1}_{n_\mathrm{p}}^\mathsf{T}]^\mathsf{T} = \mathbf{\Phi}\boldsymbol{\phi}_* \tag{7.56}$$

denotes the vector of covariances between the point to predict, $\boldsymbol{\phi}_*$, and the $n_\mathrm{tr} + n_\mathrm{p}$ training points, $\mathbf{\Phi}$.[10]

When the feature space of the kernel is finite-dimensional, i.e., the kernel is degenerate, it is possible to translate the dual formulation—in the sample space—to the primal formulation—in the feature space—so the analysis would be similar. This is the case, for instance, of the linear and polynomial kernels, and the relevance vector machine (RVM) for regression (special case of GPR),[11] which is a Bayesian sparse kernel technique that, compared to Bayesian linear regression, introduces a separate hyperparameter λ_i for each of the model's parameters instead of a single shared hyperparameter. This causes a subset of the parameters to be driven to zero.

[10] We denote a column vector of ones as $\mathbf{1}_{n_\mathrm{p}} \in \mathbb{R}^{n_\mathrm{p}}$.

[11] There is indeed an inherent conflict for the kernel: degeneracy of the kernel is good for computational reasons and it eases our theoretical analysis, but it can be bad for modelling reasons. The use of a degenerate kernel which depends on the data has some undesirable effects. As a classic example, for an RVM using localised basis functions, imagine a test point, $\mathbf{x}_*$, which lies far away from the relevance vectors. The model should not be able to draw strong conclusions about the output (we are extrapolating), but the predictive uncertainty becomes very small—this is the opposite behaviour of what we would expect from a reasonable model (Rasmussen and Williams, 2006).

In RVM for regression, due to the use of fixed basis functions, the parameters are equivalent to $\mathbf{w}^\star_{\mathrm{tr}_\lambda} = \mathbf{S}^{-1}_{\mathrm{tr}_\lambda}\boldsymbol{\Phi}^\mathsf{T}_{\mathrm{tr}_e}\mathbf{y}_{\mathrm{tr}}$, where $\mathbf{S}_{\mathrm{tr}_\lambda}$ is the regularised scatter matrix of the clean training data. The extension would be possible assuming a variance of the noise $\neq 1$, and that $\mathbf{S}_{\mathrm{tr}_\lambda} = \boldsymbol{\Phi}^\mathsf{T}_{\mathrm{tr}_e}\boldsymbol{\Phi}_{\mathrm{tr}_e} + \mathrm{Diag}(\boldsymbol{\lambda})$, where $\mathrm{Diag}(\,\cdot\,)$ denotes the diagonal matrix and $[\boldsymbol{\lambda}]_i = \lambda_i$.

If the kernel is not degenerate, the difficulty is that, due to the dual formulation of the problem, the kernel span is in the sample space, so we can no longer use a decomposition "trick" similar to (7.38) and (7.39). The key point is thus to understand the relationship between the kernel of the complete (poisoned) training data and the kernel of the clean data. For instance, one possible way to approach the problem is to use the inverse of the block matrix corresponding to the kernel of the complete (poisoned) training data. Alternatively, in the case of GPR, it could be possible to follow an online approximation to the problem through recursive Bayesian estimation, a.k.a. Bayes filter. In this respect, the updated prior distribution would correspond to the posterior distribution of the clean data, and we would need to update the posterior distribution, corresponding to the complete (poisoned) training data.

On the other hand, it is well known that an infinitely-wide DNN can be approximated by a GP model with an appropriate kernel and hyperparameters (Lee et al., 2018). Under this assumption, it could be possible to approximate the "poisonability" of an end-to-end DNN by studying the poisonability of the surrogate GP model.

Classification Models

For classification problems using sigmoid activation (softmax activation for multiclass problems) combined with cross-entropy loss function, e.g., logistic regression, the difficulty is that they generally do not admit a closed-form solution, and hence it is difficult to obtain a explicit relationship between the clean and poisoned parameters. Assuming that the loss function is convex, the learnt parameters can be decomposed as a series of updates corresponding to Newton's method. A possible initial approach to the problem could be then to compare the clean and poisoned parameters after a series of Newton's updates.

As a first approximation to the poisonability problem, in this chapter we have focussed on rank-1 attacks. That is, the poisoning points are a set of n_p identical copies of the same point—which we called poisoning prototype. In the next section we will develop a generalisation of our approach where not all the poisoning points are necessarily the same.

Because the problem is computationally challenging, we will estimate an upper bound on the degree of the polynomial for ridge regression.

7.5 On the Degree of the Polynomial for Ridge Regression

We aim to estimate an upper bound on the degree of the polynomial for ridge regression, when the poisoning points are not necessarily the same. Let us assume a poisoned ridge regression model with poisoned parameters $\mathbf{w}_\lambda^\star \in \mathbb{R}^d$ learnt on the poisoned training set $\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}}$, where $\mathcal{D}_{\text{tr}} = (\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}) = \{(\mathbf{x}_{\text{tr}_i}, y_{\text{tr}_i})\}_{i=1}^{n_{\text{tr}}}$ is IID, and regularisation hyperparameter $\lambda > 0$. Let us also assume that the poisoning points $\mathcal{D}_{\text{p}} = (\mathbf{X}_{\text{p}}, \mathbf{y}_{\text{p}}) = \{(\mathbf{x}_{\text{p}}, y_{\text{p}})\}_{i=1}^{n_{\text{p}}}$ (that follow the same distribution but different from the one for $\mathcal{D}_{\text{tr}}$) are the union of c_{p} different points, linearly independent in the basis functions space, and repeated $\frac{n_{\text{p}}}{c_{\text{p}}}$ times (n_{p} is a multiple of c_{p}). That is, $\mathcal{D}_{\text{p}} = \bigcup_{i=1}^{c_{\text{p}}} \mathcal{D}_{\text{p}_i}$ and $|\mathcal{D}_{\text{p}_i}| = \frac{n_{\text{p}}}{c_{\text{p}}}$, where $\mathcal{D}_{\text{p}_i} \neq \mathcal{D}_{\text{p}_j} \forall i \neq j$. Therefore, $\text{rank}(\boldsymbol{\Phi}_{\text{p}_e}) > 1$. We aim to derive the expression for the model's parameters $\mathbf{w}_{\lambda|c_{\text{p}}}^\star$ when poisoned by c_{p} different points, each of them repeated $\frac{n_{\text{p}}}{c_{\text{p}}}$ times.

Given $\mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{p}} = (\boldsymbol{\Phi}_e, \mathbf{y})$, $\mathcal{D}_{\text{tr}} = (\boldsymbol{\Phi}_{\text{tr}_e}, \mathbf{y}_{\text{tr}})$, $\mathcal{D}_{\text{p}} = (\boldsymbol{\Phi}_{\text{p}_e}, \mathbf{y}_{\text{p}})$, and let $\mathbf{S}$, $\mathbf{S}_{\text{tr}}$ and $\mathbf{S}_{\text{p}}$ be the scatter matrices of the complete training, clean training and poisoning set, respectively, and $\boldsymbol{\Phi}_e$, $\boldsymbol{\Phi}_{\text{tr}_e}$ and $\boldsymbol{\Phi}_{\text{p}_e}$ the extended design matrices of the complete training, clean training and poisoning set, respectively. As mentioned in Subsect. 7.3.3, we shall proceed with a continuous relaxation of the problem, so we define the real variables greater than or equal to one: $\hat{n}_{\text{p}_\lambda}^\star, \hat{n}_{\text{p}_\lambda} \in \mathbb{R}_{\geq 1}$.

We therefore have the following objective:

$$p(\hat{n}_{\text{p}_\lambda}, \xi_{\text{p}}) = \left(c_{\text{p}}^{-1} \sum_{i=1}^{c_{\text{p}}} \left(y_{\text{p}_i} - \boldsymbol{\phi}_{\text{p}_{e_i}}^\intercal \mathbf{w}_{\lambda|c_{\text{p}}}^\star \right)^2 - \xi_{\text{p}} n_{\text{tr}}^{-1} ||\mathbf{y}_{\text{tr}} - \boldsymbol{\Phi}_{\text{tr}_e} \mathbf{w}_{\lambda|c_{\text{p}}}^\star||_2^2 \right) \mathbb{1}_{\hat{n}_{\text{p}_\lambda} \geq 1}(\hat{n}_{\text{p}_\lambda}) < 0, \tag{7.57}$$

which is also a truncated polynomial. When the degree of the polynomial in the objective is greater than two, which is the case when the $c_{\text{p}} > 1$, analytically computing its roots is tedious, not easy to interpret and often intractable. These roots can be computed with root-finding algorithms—standard numerical methods—such as Newton's method, bisection method, secant method, or the QR algorithm to compute the eigenvalues of the Companion matrix corresponding to the polynomial.

7.5.1 Characterising the Polynomial in the Objective Function

Rather than computing the exact value of all of its roots, there are other characteristics of the polynomial to determine that are relevant to the poisonability problem. In particular: (i) an upper bound on the *degree* of the polynomial, which effectively provides an upper bound on the number of breaking points of the model; (ii) an upper bound on the value of the positive real roots, which provides the maximum value of the breaking points; and (iii) the total number of distinct real roots greater than or equal to one of the polynomial, that is, the actual number of breaking points. The upper bound on the value of the positive real roots, out of the scope of this thesis, can be determined, e.g, using Cauchy's bound (Cauchy, 1829), Kioustelidis's bounds (Kioustelidis, 1986), or Akritas, Strzeboński, and Vigklas's bounds (Akritas et al., 2006). The total number of distinct real roots greater than or equal to one can be calculated by applying Sturm's theorem (Sturm, 1829) in $p(\hat{n}_{\mathrm{p}_\lambda})$ in the interval $\hat{n}_{\mathrm{p}_\lambda} \in (1, \infty)$; an upper bound can also be obtained by applying Descartes' rule of signs in $p(\hat{n}_{\mathrm{p}_\lambda} + 1)$.

We shall focus on estimating the degree of the polynomial. The first step is to reduce (7.57) to its largest terms:

$$\left(c_{\mathrm{p}}^{-1} \sum_{i=1}^{c_{\mathrm{p}}} \left(\boldsymbol{\phi}_{\mathrm{p}_{e_i}}^{\mathsf{T}} \mathbf{w}_{\lambda|c_{\mathrm{p}}}^{\star} \right)^2 - \xi_{\mathrm{p}} n_{\mathrm{tr}}^{-1} ||\boldsymbol{\Phi}_{\mathrm{tr}_e} \mathbf{w}_{\lambda|c_{\mathrm{p}}}^{\star}||_2^2 \right) \mathbb{1}_{\hat{n}_{\mathrm{p}_\lambda} \geq 1}(\hat{n}_{\mathrm{p}_\lambda}) < 0. \tag{7.58}$$

The second step is to decompose the regularised scatter matrix $\mathbf{S}_\lambda$ of the complete training set as

$$\mathbf{S}_\lambda = \mathbf{S} + \lambda \mathbf{I}_{d_e} = \mathbf{S}_{\mathrm{tr}} + \lambda \mathbf{I}_{d_e} + \mathbf{S}_{\mathrm{p}} = \mathbf{S}_{\mathrm{tr}} + \lambda \mathbf{I}_{d_e} + \frac{\hat{n}_{\mathrm{p}_\lambda}}{c_{\mathrm{p}}} \sum_{i=1}^{c_{\mathrm{p}}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^{\mathsf{T}}, \tag{7.59}$$

where $\boldsymbol{\phi}_{\mathrm{p}_{e_i}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^{\mathsf{T}}$ is the outer product of the poisoning point $\boldsymbol{\phi}_{\mathrm{p}_{e_i}}$ with itself. Similarly, we can write

$$\boldsymbol{\Phi}^{\mathsf{T}} \mathbf{y} = \boldsymbol{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}} \mathbf{y}_{\mathrm{tr}} + \boldsymbol{\Phi}_{\mathrm{p}_e}^{\mathsf{T}} \mathbf{y}_{\mathrm{p}} = \boldsymbol{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}} \mathbf{y}_{\mathrm{tr}} + \frac{\hat{n}_{\mathrm{p}_\lambda}}{c_{\mathrm{p}}} \sum_{i=1}^{c_{\mathrm{p}}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}} y_{\mathrm{p}_i}. \tag{7.60}$$

Thus, the expression for $\mathbf{w}_{\lambda|c_{\mathrm{p}}}^{\star}$ is:

$$\mathbf{w}_{\lambda|c_{\mathrm{p}}}^{\star} = \mathbf{S}_\lambda^{-1} \boldsymbol{\Phi}_e^{\mathsf{T}} \mathbf{y} = \left(\mathbf{S}_{\mathrm{tr}} + \lambda \mathbf{I}_{d_e} + \frac{\hat{n}_{\mathrm{p}_\lambda}}{c_{\mathrm{p}}} \sum_{i=1}^{c_{\mathrm{p}}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^{\mathsf{T}} \right)^{-1} \left(\boldsymbol{\Phi}_{\mathrm{tr}_e}^{\mathsf{T}} \mathbf{y}_{\mathrm{tr}} + \frac{\hat{n}_{\mathrm{p}_\lambda}}{c_{\mathrm{p}}} \sum_{i=1}^{c_{\mathrm{p}}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}} y_{\mathrm{p}_i} \right). \tag{7.61}$$

To compute $\mathbf{S}_\lambda^{-1} = \mathbf{S}_{\lambda_{c_{\mathrm{p}}}}^{-1}$, we can recursively use the Sherman-Morrison formula

(Appx. A.2) to obtain the rank-1 updates:

$$\mathbf{S}_{\lambda_i}^{-1} = \left(\mathbf{S}_{\lambda_{i-1}} + \frac{\hat{n}_{\mathrm{p}_\lambda}}{c_\mathrm{p}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^\mathsf{T}\right)^{-1} = \mathbf{S}_{\lambda_{i-1}}^{-1} - \frac{\frac{\hat{n}_{\mathrm{p}_\lambda}}{c_\mathrm{p}} \mathbf{S}_{\lambda_{i-1}}^{-1} \boldsymbol{\phi}_{\mathrm{p}_{e_i}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^\mathsf{T} \mathbf{S}_{\lambda_{i-1}}^{-1}}{1 + \frac{\hat{n}_{\mathrm{p}_\lambda}}{c_\mathrm{p}} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}^\mathsf{T} \mathbf{S}_{\lambda_{i-1}}^{-1} \boldsymbol{\phi}_{\mathrm{p}_{e_i}}}, \; i = 1, \ldots, c_\mathrm{p}, \tag{7.62}$$

assuming that $\mathbf{S}_{\lambda_0} = \mathbf{S}_\mathrm{tr} + \lambda \mathbf{I}_{d_e}$ is non-singular. Thus, it is clear that each entry of $\mathbf{S}_\lambda^{-1}$, and therefore each entry of $\mathbf{w}_{\lambda|c_\mathrm{p}}^\star$, is a rational function of $\hat{n}_{\mathrm{p}_\lambda}$. Beyond the scope of this study, this "trick" can also help evaluate the impact of a new poisoning point on the model's parameters when we have already computed the scatter matrix corresponding to the previous poisoning points, without recomputing the inverse of the regularised scatter matrix of the whole training set.

7.5.2 Upper Bound on the Degree of the Polynomial

From (7.58), (7.61), and (7.62), we can deduce that an upper bound on the degree of the polynomial (7.57), i.e., on the maximum number of breaking points, $|\mathcal{N}_\mathrm{break}^\star|$, is two times the degree of the rational function[12] of each entry of $\mathbf{w}_{\lambda|c_\mathrm{p}}^\star$. Moreover, we already know that $|\mathcal{N}_\mathrm{break}^\star| \in \{1, 2\}$ for $c_\mathrm{p} = 1$, as analysed in Sect. 7.4. Then, by induction, we obtain the upper bound:

$$|\mathcal{N}_\mathrm{break}^\star| \leq \begin{cases} 2 & \text{if } c_\mathrm{p} = 1, \\ 2^{c_\mathrm{p}+1} & \text{if } c_\mathrm{p} \geq 2. \end{cases} \tag{7.63}$$

Even though the degree of the polynomial can appear to be large, in practice the number of breaking points can be much lower. In particular, the leading term of the polynomial may cancel out, some roots may be infeasible for the objective (7.57),[13] some roots may be very close to each other, and some roots may correspond to the same integer number. The calculation of tighter bounds on the number of breaking points is thus a potential direction to follow in the future.

Furthermore, we conjecture that the coefficient of the leading term (largest degree), a.k.a. its largest coefficient, is negative, so that when $\hat{n}_\mathrm{p} \to \infty$, $p(\hat{n}_{\mathrm{p}_\lambda}) \to -\infty$ and hence the tolerable poisoning distribution dominates the model. The intuition is as follows: when $\hat{n}_\mathrm{p}$ is very large, the terms corresponding to the largest degree dominate the polynomial,

[12] Here, the degree of the rational function is the maximum of the degrees of its constituent polynomials, when the fraction is reduced to lowest terms.

[13] For example, it is very likely that almost one half of the roots of the polynomial are negative, as seen for $c_\mathrm{p} = 1$.

as in (7.58). Then, if the poisoning points and the clean training points follow a different distribution, and there are many more poisoning points than clean training points, the parameters $\mathbf{w}^{\star}_{\lambda|c_\mathrm{p}}$ are already shifted towards the tolerable poisoning distribution. Therefore, on average, the model's mean error on the clean training data is larger than the mean error on the poisoning points, which satisfies (7.58), leading to a negative leading term.

7.6 Conclusions

Previous works in data poisoning attacks against regression algorithms assumed that the error of the poisoning points on the poisoned model is larger than the error of the clean points on the poisoned model. This may be the case if the number of poisoning points is low and/or they are located far from the clean distribution, as we observed in the experiments presented in Ch. 5, Sect. 5.5. We challenged this assumption and introduced the concept of poisonability, poisonability intolerance, and breaking points of models and datasets.

The concept of poisonability allows us to find the breaking points—in terms of the number of poisoning points needed—so that the mean error of the clean points on the poisoned model matches the tolerable mean error of the poisoning points on the poisoned model. In the poisonability problem, one of the key steps is finding a mapping between the clean and poisoned parameters—or hypothesis function—of a model. This can help understand the consequences of the attack and define standards of performance degradation with respect to a clean reference model.

In particular, we analysed the poisonability of linear regression models. Under a rank-1 attack, we theoretically found that a non-regularised linear regression model has a single breaking point; and a ridge regression model poisoned can have up to two breaking points.

Moreover, we generalised the previous result and obtained an upper bound on the degree of the objective function of the poisonability problem corresponding to ridge regression. Specifically, we analysed an attack of c_p different poisoning points (rank-c_p attack). In this case, the degree of the polynomial grows in $\mathcal{O}(2^{c_\mathrm{p}})$. This work paves the way for root-finding algorithms that can find the precise roots of the polynomial, and therefore the breaking points of the model when the poisoning points are more diverse.

Chapter 8

Conclusions

8.1 Summary of Thesis Achievements and their Implications

In this thesis, we have made progress towards understanding and mitigating indiscriminate data poisoning attacks in worst-case scenarios against supervised learning, from two perspectives: (i) considering the ML pipeline: data sanitisation, hyperparameter learning, and model training; and (ii) novel insights into regression poisoning.

In particular, in Ch. 3 we showed that current approaches in classification settings, which typically assume that hyperparameters remain constant, lead to an overly pessimistic view of the algorithms' robustness and of the impact of regularisation. We proposed a novel optimal attack formulation that considers the effect of the attack on the hyperparameters and models the attack as a *multiobjective* bilevel optimisation problem. This allows to formulate optimal attacks, learn hyperparameters and evaluate robustness under worst-case conditions. We applied this attack formulation to several ML classifiers using L_2 and L_1 regularisation. Our evaluation on multiple datasets shows that choosing an "a priori" constant value for the regularisation hyperparameter can be detrimental to the performance of the algorithms. This confirms the limitations of previous strategies and evidences the benefits of using L_2 and L_1 regularisation to dampen the effect of poisoning attacks, when hyperparameters are learnt using a small clean trusted dataset. When the regularisation

hyperparameter is learnt appropriately, the algorithm is more robust and, at the same time, the performance of the model is not affected when there is no attack. The mitigating effect of L_2 regularisation was also supplemented by the theoretical results obtained in Ch. 6 in the context of ridge regression. Additionally, our results show that the use of regularisation plays an important robustness and stability role in complex models, such as DNNs, where the attacker can have more flexibility to manipulate the decision boundary. In summary, this is an important step towards understanding the trade-off between accuracy and robustness to a poisoning attack by selecting an appropriate value of the regularisation hyperparameter. Although acquiring a trusted set can be expensive and often require human inspection, this investment can pay off if most of the training data is collected from untrusted sources. Therefore, a trusted set is important to guide the learning process of the hyperparameters: if this set is poisoned, the attacker can also corrupt the hyperparameters learnt. There is still room for investigating the best uses of a trusted validation set: both for data sanitisation and for model and hyperparameter selection. Moreover, this contribution can open new research directions towards attacks that target the hyperparameters and/or the selection of a particular model.

We then moved on to analysing the vulnerabilities of regression models under data poisoning attacks, which have been somehow overlooked by the adversarial ML community. This allowed us to introduce a much broader set of vulnerabilities that may arise in regression settings, compared to classification models. Combined with their widespread use in mission-critical systems, this makes regression models a prime and vulnerable target for poisoning attacks. Specifically, we introduced in Ch. 4 a novel threat model for poisoning attacks against regression models. To our knowledge, there was no comprehensive threat model of data poisoning attacks against regression models. We evaluated several aspects of data poisoning attacks against regression learning, exceeding previous threat models both in terms of breadth and depth. Attacks against regression algorithms generalise the ones against classification (in terms of the attacker's objective), since the labels in the former are continuous. We presented a large variety of error-specific attacks, including, for instance, assumptions violation, wrong model selection, interpretability violation, and uncertainty fluctuation. Moreover, we theoretically showed that in linear regression models, any manipulation of a poisoning point directly affects all the parameters of the model. This result limits the possibility of creating very targeted attacks or backdoors in linear models, since one of the requirements of a backdoor attack is that the model should behave

correctly under clean samples. We also theoretically showed that the feasible poisoning error increases with the "natural" noise level of the dataset and with the mismatch between the estimated and the true mapping function of the model. This result suggests that datasets that are more difficult to fit to are also more difficult to defend when poisoned.

Equipped with this threat model, in Ch. 5 we emphasised that the robustness of regression models to poisoning attacks has not been widely studied in the literature. Most of works that do address it assume that the attacker's objective is to maximise the damage caused by the attack, without properly considering detectability constraints that can help evade defences. For this reason, we proposed a novel stealthy attack formulation against regression algorithms via multiobjective bilevel optimisation: one of the objectives accounts for attack effectiveness, while the other accounts for the detectability. This formulation is more realistic as in many applications users will deploy a detector to reject points that are suspicious. We also proposed a novel detectability-risk function and a novel approach to solving the multiobjective bilevel optimisation problem via normalisation of the objectives. This allows us to consider different degrees of detectability for the attacker. Our experiments in both linear regression models and DNNs showed that state-of-the-art defences do not mitigate these stealthy attacks, even if they are not designed as adaptive attacks targeting those specific defences—in other words, without detectability constraints. Moreover, to our knowledge, we are the first to analyse optimal data poisoning attacks targeting DNNs in regression settings. We evaluated different defences proposed so far on real-world datasets and identified the factors that influence their effectiveness such as the use of a trusted clean set. We empirically confirmed our hypothesis that datasets that are more difficult for the ML model to fit to (e.g., because they are noisy), are also more difficult to defend. Specifically, this supports the theoretical findings shown in Ch. 4, Sect. 4.4. We expect that these contributions will be of interest to both the adversarial ML and the ML optimisation communities. In the adversarial ML community, the lessons learnt evidence the need for specific defences in the regression domain, and defences that also consider stealthy attackers, which is a challenging problem since it is often difficult to distinguish between genuine and malicious samples in those cases. Moreover, attacks that consider the continuous manipulation of the poisoning labels can lead to new attack formulations with detectability with different goals in mind. In the ML optimisation community, we technically contribute to solving multiobjective bilevel optimisation problems and bring to light some of its challenges when the objectives are of different orders of magnitude.

In Ch. 5 we also proposed BayesClean, a novel defence built upon a Bayesian linear regression model, that improves on state-of-the-art defences when attacks consider detectability constraints and when the number of poisoning points is large. At a high level, the defence rejects poisoning points outside of a certain "confidence interval" around the mean prediction of the Bayesian linear regression model. We empirically showed the effectiveness of BayesClean to mitigate strong and stealthy attacks. In contrast to previous defences looking at the errors or the norm of the gradients of the training samples, this shows a promising research direction considering the model uncertainty, and the need to develop techniques that account for this property to counter poisoning attacks.

To understand the challenges and limitations of the poisoning attack formulations proposed against regression models, in Ch. 6 we theoretically analysed some fundamental properties of the attacker's objective functions. To the best of our knowledge, we are the first to thoroughly characterise indiscriminate data poisoning attacks against ridge regression. We showed that, without considering detectability constraints, the attacker's objective is convex in the poisoning labels, but non-convex in the poisoning features (or basis functions if used), and therefore it is non-convex when combining both set of variables. In particular, given the stationarity conditions for the basis functions, we conjectured that most of the critical points (namely stationary points) of the attacker's objective are saddle points. This justifies the normalisation of hypergradients used in the experiments of Ch. 5, to help escape the saddle points and accelerate the optimisation process. With regard to the asymptotic behaviour of the attacker's objective, we theoretically determined the intuitive result that the attacker is generally unbounded if the poisoning basis functions and/or labels are unbounded. However, regularising the model can prevent the attacker from being unbounded, which is coherent with the results obtained in Ch. 3 in classification settings.

In Ch. 6 we also showed the equivalence between the optimal indiscriminate attack against ridge regression and CQP (if only the poisoning labels are optimised), and obtained some bounds on the rank of the hyper-Hessian matrix—Hessian matrix of the attacker's objective function: under certain assumptions, the hyper-Hessian is non-singular if and only if all the poisoning points are linearly independent in the basis functions (or feature) space. Therefore, optimising more poisoning points than the number of basis functions may be redundant. When considering detectability constraints, we showed that the attacker's

objective's convexity can be brittle: because the attacker's objective is a sum of a convex objective—effectiveness—and a concave objective—detectability—the objective function can undesirably shift from convexity to concavity, depending on the values of their respective hyper-Hessians and the hyperparameter that controls their trade-off. We also defined the attacker's multiobjective condition number, which is the ratio between the largest eigenvalue of the hyper-Hessian of the effectiveness objective, and the smallest eigenvalue of the hyper-Hessian of the detectability objective. In practice, a very large value of the condition number leads to an objective function predominantly convex—the poisoning labels lie in the boundary of the feasible set. In contrast, a very low value of the condition number means that the objective is predominantly concave—the poisoning labels lie in the local minimum of the detectability-risk function, if the minimum is in the interior of the feasible set, or at its boundary otherwise. We also obtained a relationship between α and the condition number that determines the concavity of the objective. Finally, the lessons learnt allowed us to develop the attacker's KKT conditions, and to discuss the different solutions depending on the convexity of the attacker's objective. We also justified our proposed detectability-risk function in Ch. 5, based on a product of two convex functions, which can provide more diversity of poisoning points—and potentially increasing the difficulty to detect them. If the effectiveness objective is convex and quadratic, and the detectability-risk function is non-negative, then a quartic detectability-risk function can provide diversity for the optimal poisoning points.

Finally, we challenged the underlying assumption of most defences in regression settings (the error of the poisoning points is larger than the error of the clean points) and introduced the concept of "poisonability" of models and datasets in Ch. 7. The poisonability allows us to find the number of poisoning points required so that the mean error of the clean points matches the tolerable mean error of the poisoning points on the poisoned model. In particular, we analysed the poisonability of linear regression models. Under a rank-1 attack, we theoretically found the breaking points of both non-regularised linear regression and ridge regression models. To our knowledge, the concept of poisonability is new in the data poisoning literature. Our definitions and analysis can foster future research on understanding the consequences of attacks, explaining why some defences looking at the error of the training data fail, and defining standards of performance degradation with respect to a clean reference model.

We summarise our achievements in Thesis Achievements and in Fig. 8.1.

Thesis Achievements

- A novel optimal attack formulation that considers the interactions between the attack and the model's hyperparameters.
- Regularisation, together with a small trusted set, as a practical methodology to help mitigate poisoning attacks and obtain the best trade-off between accuracy and robustness.
- A novel threat model of poisoning attacks against regression models.
- A novel optimal attack formulation capable of modelling different trade-offs between effectiveness and detectability in regression settings.
- A novel defence, based on the model uncertainty, to mitigate strong and stealthy attacks.
- A theoretical analysis of optimal attacks against ridge regression models, showing the challenges and limitations of the attack formulations presented.
- A characterisation of the "poisonability" of linear regression models: for ridge regression, we determine the number of poisoning points needed so that the mean error of the poisoning points matches the mean error of the clean points on the poisoned model.

8.2 Future Work

There are several challenges to address in the future. Here we summarise some potential avenues for further work.

Data poisoning, regularisation, and defences: we plan to investigate the effect of regularisation in targeted poisoning attacks—which can be more difficult to mitigate since they have a lower impact on the norm of the model's parameters—and ways to combine and contrast the mitigating effect obtained from regularisation with that of other defences against poisoning attacks, e.g., data sanitisation.

Model selection under data poisoning: a potential area that has been less explored in the literature lies at the intersection between model selection and data poisoning: from the attacker's perspective, it is possible to propose attacks that can target a particular hyperparameter. For instance, an attacker could force the learner to select a large degree for the hypothesis function. From the defender's viewpoint, an interesting research direction is

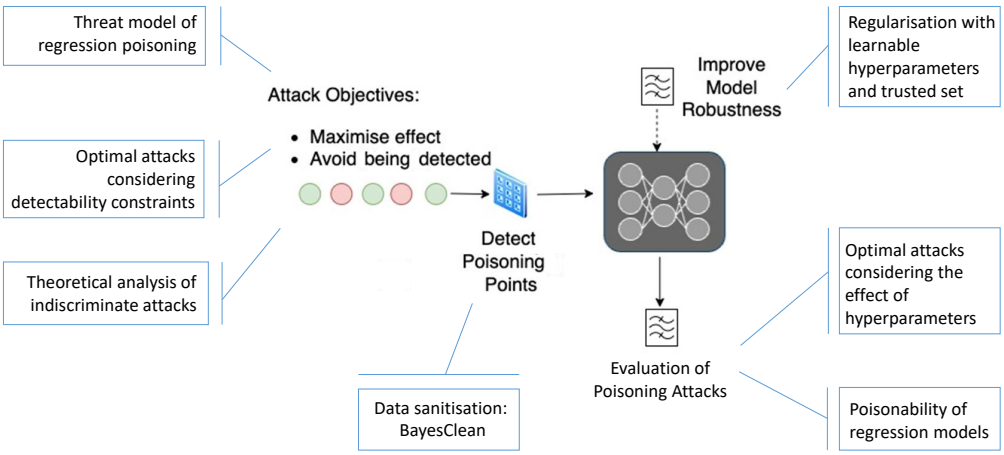


Figure 8.1: Main achievements of this thesis considering the ML pipeline.

the design of models and architectures that can be more robust to poisoning attacks—e.g., by leveraging differentiable architecture search (Liu et al., 2019) under poisoning.

Error-specific attacks against regression models: beyond the MSE, many attacker's goals can be explored. For instance, maximising or minimising the bias of the model, forcing some parameters to be close zero—which, in the case of linear regression, also relates to the interpretability of the model—or forcing the labels to follow a particular function.

Detectability-risk function: in Ch. 5 the detectability-risk function of the attack was based on the effect it has on the loss function of the model. This is an appropriate choice as a defender is likely to discard points that have undue effects. However, it is also possible, to investigate different formulations of the detectability, e.g., when the model's parameters used as a reference are poisoned; and also based on the characteristics of the dataset.

Data poisoning attacks against Gaussian process regression and model uncertainty: a natural extension of our work is the formulation of optimal poisoning attacks targeting the posterior predictive distribution of GPR (Rasmussen and Williams, 2006). GPs are well-known Bayesian learning methods that have been shown to provide state-of-the-art performance in complex regression learning tasks. In addition to being

highly accurate, they also present a number of other appealing features, such as probabilistic predictions and a simple model selection scheme. GPs fall under kernel methods, and are especially useful for low data regime. Because GPs are non-parametric, the training points are explicitly needed to make inference at test time. This distinction, together with the uncertainty that naturally appears in the model's predictions, makes the problem interesting to analyse. The analysis of data poisoning attacks against GPR can thus provide new insights to understand the synergies between the model uncertainty and the poisoning attack. Moreover, well-calibrated uncertainty estimates improve transparency for ML models (Bhatt et al., 2021). Additional questions to understand are which kernels are more robust to poisoning perturbations, and whether GPs' scalability and robustness to poisoning attacks are at odds. Then, it can also be possible to target robust GPR models (Kuss, 2006; Jylänki et al., 2011), which acknowledge the existence of outliers, and can allow to naturally model detectability constraints.[1]

Analysis of poisoning attacks through matrix perturbation theory: for the theoretical analysis of attacks against regression models, future work will incorporate matrix perturbation theory (Wei, 1990; Li, 2006; Osborne, 2010), to understand the bounds on the variations of the model's parameters and the mean squared errors, for small poisoning perturbations and attacks of different rank.

Poisonability and robustness: from a robustness standpoint, a desirable aspect of a model and dataset is to find the conditions in which it is more difficult to poison them, i.e., when the value of the maximum breaking point increases. A possible approach is to search for the optimal value of the regularisation hyperparameter can help increase the value of breaking point of the model. This problem can provide a different view on the robustness of ridge regression to poisoning, which we aim to tackle in the future.

Poisonability and summary points: as mentioned in Subsect. 7.3.2, without additional assumptions, finding a mapping between the clean and poisoned parameters is computationally challenging, because it requires to enumerate all possible values of the poisoning points, which is an infinite set. Therefore, further assumptions for the sample space over the poisoning dataset are needed to make the problem tractable. In particular, given a poisoned target model and a clean training set, a problem that deserves further research is: what is the minimum number of poisoning points needed to characterise a

[1]This could be possible, for instance, through a likelihood composed of a mixture of a model which accounts for the regular observations and a second model for explaining outliers.

poisoning attack, i.e., that can attain the target model? This problem is especially relevant from a poisonability perspective. This line of research was also previously started by (Koh et al., 2022), and it is related to the teaching dimension of a model (Liu et al., 2016; Zhu et al., 2018; Kumar et al., 2021; Qian et al., 2022). There may also exist some connections with sample compression (Floyd and Warmuth, 1995; Doliwa et al., 2014).

Expanding the poisonability analysis: there are many avenues to expand our poisonability analysis. One of them is classification models: the difficulty is that they generally do not admit a closed-form solution, and hence it is difficult to obtain a explicit relationship between the clean and poisoned parameters. Assuming that the loss function is convex, the learnt parameters can be decomposed as a series of updates corresponding to Newton's method. A possible first approximation to the problem would be then to compare the clean and poisoned parameters after a series of Newton's updates. On the other hand, the analysis can also be extended to kernel methods: if the kernel is not degenerate, the difficulty is that, due to the dual formulation of the problem, the kernel span is in the sample space, so we can no longer use a decomposition trick similar to the one of Ch. 7. The key point is thus to understand the relationship between the kernel of the complete (poisoned) training data and the kernel of the clean data. For instance, one possible way to approach the problem is to use the inverse of the block matrix corresponding to the kernel of the complete (poisoned) training data. Alternatively, in the case of GPR, it can be possible to follow an online approximation to the problem through recursive Bayesian estimation, a.k.a. Bayes filter. In this respect, the updated prior distribution would correspond to the posterior distribution of the clean data, and we would need to update the posterior distribution, corresponding to the complete (poisoned) training data.

Data poisoning attacks in online learning settings: real-world ML systems often update in an online fashion. For example, in e-commerce applications, user-generated data arrives sequentially, and the learner usually updates daily to learn from the newly-acquired data. Thus, online learning settings represent natural scenarios where data poisoning can arise. The main challenges are that, in online settings, the attacker can only observe the current training data, and is required to make a sequence of decisions to perform the attack through time (Wang and Chaudhuri, 2018; Collinge et al., 2019; Zhang et al., 2020). One of the key aspects that are not well understood is how to distinguish between poisoning attack and a genuine concept drift, where the trade-off between stability and plasticity of

the learning algorithm plays a crucial role.

Data poisoning attacks against reinforcement learning: given that RL algorithms are increasingly used in critical applications, including cyber-physical systems (Li and Qiu, 2019), personal assistive devices (Rybski et al., 2007), or large language models (Christiano et al., 2017; Ouyang et al., 2022), it is of utmost importance to investigate the security threat to RL algorithms against different forms of data poisoning attacks. The majority of existing works (Huang et al., 2017a; Kos and Song, 2017; Lin et al., 2017) focus on evasion attacks against RL agents, challenging an already learnt and fixed policy. On the other hand, data poisoning attacks against RL agents can generally influence both the reward feedback given to the agent and the state to which the agent transitions, i.e., its transition dynamics. For instance, an attacker can have a target policy in mind and poison the environment with the goal of teaching/forcing the RL agent to executing this policy. In many practical scenarios, e.g., chatbots, both rewards and transitions can be indirectly poisoned through interaction data (Rakhsha et al., 2021). Therefore, the paradigm of RL opens up a new family of attack formulations, in which the development of robust techniques and robust Markov decision processes (Lim et al., 2013) is of crucial importance. Modelling the possible threats in these environments and understanding the poisonability of RL models are therefore important steps towards the safe and secure deployment of RL.

Bibliography

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep Learning with Differential Privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.

Aghakhani, H., Meng, D., Wang, Y.-X., Kruegel, C., and Vigna, G. (2021). Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability. In *IEEE European Symposium on Security and Privacy*, pages 159–178.

Akazaki, T. (2016). Falsification of Conditional Safety Properties for Cyber-Physical Systems with Gaussian Process Regression. In *International Conference on Runtime Verification*, pages 439–446. Springer.

Akritas, A., Strzeboński, A., and Vigklas, P. (2006). Implementations of a New Theorem for Computing Bounds for Positive Roots of Polynomials. *Computing*, 78:355–367.

Aksen, D. and Aras, N. (2012). A bilevel fixed charge location model for facilities under imminent attack. *Computers & Operations Research*, 39(7):1364–1381.

Alfeld, S., Vartanian, A., Newman-Johnson, L., and Rubinstein, B. I. P. (2019). Attacking Data Transforming Learners at Training Time. In *AAAI Conference on Artificial Intelligence*, pages 3167–3174.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020). How To Backdoor

Federated Learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948.

Banks, A. and Ashmore, R. (2019). Requirements Assurance in Machine Learning. In *AAAI Workshop on Artificial Intelligence Safety*.

Bard, J. F. (2013). *Practical Bilevel Optimization: Algorithms and Applications*, volume 30. Springer Science & Business Media.

Barreno, M., Nelson, B., Joseph, A. D., and Tygar, J. D. (2010). The security of machine learning. *Machine Learning*, 81(2):121–148.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. (2006). Can Machine Learning Be Secure? In *ACM Symposium on Information, Computer and Communications Security*, pages 16–25.

Bauer, H. (1958). Minimalstellen von Funktionen und Extremalpunkte. *Archiv der Mathematik*, 9(4):389–393.

Belsley, D. A., Kuh, E., and Welsch, R. E. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley & Sons.

Bengio, Y. (2000a). Continuous optimization of hyper-parameters. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks. Neural Computing: New Challenges and Perspectives for the New Millennium*, pages 305–310.

Bengio, Y. (2000b). Gradient-Based Optimization of Hyperparameters. *Neural Computation*, 12(8):1889–1900.

Bernardo, J. M. and Smith, A. F. M. (2009). *Bayesian Theory*, volume 405. John Wiley & Sons.

Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G., Krishnan, R., Stanley, J., Tickoo, O., et al. (2021). Uncertainty as a Form of Transparency: Measuring, Communicating, and Using Uncertainty. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 401–413.

Biggio, B., Corona, I., Fumera, G., Giacinto, G., and Roli, F. (2011a). Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks. In *International Workshop on Multiple Classifier Systems*, pages 350–359. Springer.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion Attacks against Machine Learning at Test Time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402.

Biggio, B., Fumera, G., and Roli, F. (2014). Security Evaluation of Pattern Classifiers under Attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996.

Biggio, B., Nelson, B., and Laskov, P. (2011b). Support Vector Machines Under Adversarial Label Noise. In *Asian Conference on Machine Learning*, pages 97–112.

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning Attacks against Support Vector Machines. In *International Conference on Machine Learning*, pages 1467–1474.

Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bishop, C. M. and Bishop, H. (2024). *Deep Learning: Foundations and Concepts*. Springer.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987). Occam's razor. *Information processing letters*, 24(6):377–380.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*.

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bracken, J. and McGill, J. T. (1973). Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 21(1):37–44.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24:123–140.

Brown, G., Carlyle, M., Diehl, D., Kline, J., and Wood, K. (2005). A Two-Sided Optimization for Theater Ballistic Missile Defense. *Operations Research*, 53(5):745–763.

Brown, G. G., Carlyle, W. M., Harney, R. C., Skroch, E. M., and Wood, R. K. (2009). Interdicting a nuclear-weapons project. *Operations Research*, 57(4):866–877.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901.

Brunsdon, C., Fotheringham, A. S., and Charlton, M. E. (1996). Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity. *Geographical Analysis*, 28(4):281–298.

Candanedo, L. (2017). Appliances energy prediction. [Online]. Available from: `https://doi.org/10.24432/C5VC8G`. [Accessed: 10/11/2023].

Carlini, N., Jagielski, M., Choquette-Choo, C. A., Paleka, D., Pearce, W., Anderson, H., Terzis, A., Thomas, K., and Tramèr, F. (2023). Poisoning Web-Scale Training Datasets is Practicall. *arXiv preprint arXiv:2302.10149*.

Carnerero-Cano, J., Muñoz-González, L., Spencer, P., and Lupu, E. C. (2020). Regularisation Can Mitigate Poisoning Attacks: A Novel Analysis Based on Multiobjective Bilevel Optimisation. *arXiv preprint arXiv:2003.00040*.

Carnerero-Cano, J., Muñoz-González, L., Spencer, P., and Lupu, E. C. (2021). Regularization Can Help Mitigate Poisoning Attacks... with the Right Hyperparameters. In *ICLR Workshop on Security and Safety in Machine Learning Systems*.

Carnerero-Cano, J., Muñoz-González, L., Spencer, P., and Lupu, E. C. (2023). Hyperparameter Learning under Data Poisoning: Analysis of the Influence of Regularization via Multiobjective Bilevel Optimization. *IEEE Transactions on Neural Networks and Learning Systems*.

Cauchy, A.-L. (1829). Exercices de mathématique. *Oeuvres*, 9(2):122.

Chati, Y. S. and Balakrishnan, H. (2017). A Gaussian Process Regression Approach to Model Aircraft Engine Fuel Flow Rate. In *International Conference on Cyber-Physical Systems*, pages 131–140.

Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I.,

and Srivastava, B. (2019). Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *AAAI Workshop on Artificial Intelligence Safety*.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017a). ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *ACM Workshop on Artificial Intelligence and Security*, pages 15–26.

Chen, R., Li, Z., Li, J., Yan, J., and Wu, C. (2022). On Collective Robustness of Bagging Against Data Poisoning. In *International Conference on Machine Learning*, pages 3299–3319.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. (2017b). Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv preprint arXiv:1712.05526*.

Cheng, Z., Liang, J., Choi, H., Tao, G., Cao, Z., Liu, D., and Zhang, X. (2022). Physical Attack on Monocular Depth Estimation with Optimal Adversarial Patches. In *European Conference on Computer Vision*, pages 514–532. Springer.

Chinchuluun, A., Pardalos, P. M., and Enkhbat, R. (2005). Global minimization algorithms for concave quadratic programming problems. *Optimization*, 54(6):627–639.

Chio, C. and Freeman, D. (2018). *Machine Learning and Security: Protecting Systems with Data and Algorithms*. O'Reilly Media, Inc.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, pages 4302–4310.

Cinà, A. E., Demontis, A., Biggio, B., Roli, F., and Pelillo, M. (2022). Energy-Latency Attacks via Sponge Poisoning. *arXiv preprint arXiv:2203.08147*.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., and Roli, F. (2023). Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning. *ACM Computing Surveys*, 55(13s):1–39.

Cinà, A. E., Vascon, S., Demontis, A., Biggio, B., Roli, F., and Pelillo, M. (2021). The

Hammer and the Nut: Is Bilevel Optimization Really Needed to Poison Linear Classifiers? In *IEEE International Joint Conference on Neural Networks*, pages 1–8.

Clark, P. A. and Westerberg, A. W. (1983). Optimization for design problems having more than one objective. *Computers & Chemical Engineering*, 7(4):259–278.

Cline, A. K. and Dhillon, I. S. (2006). Computation of the Singular Value Decomposition. In *Handbook of Linear Algebra*, pages 45–1–45–13. Chapman and Hall/CRC.

Co, K. T., Muñoz-González, L., de Maupeou, S., and Lupu, E. C. (2019). Procedural Noise Adversarial Examples for Black-Box Attacks on Deep Convolutional Networks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 275–289.

Collinge, G., Lupu, E. C., and Muñoz-González, L. (2019). Defending against Poisoning Attacks in Online Learning Settings. In *European Symposium on Artificial Neural Networks*.

Colson, B., Marcotte, P., and Savard, G. (2005). Bilevel programming: A survey. *4or*, 3(2):87–107.

Colson, B., Marcotte, P., and Savard, G. (2007). An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256.

Constantin, L. (2021). How data poisoning attacks corrupt machine learning models. [Online]. Available from: `https://www.csoonline.com/article/570555/how-data-poisoning-attacks-corrupt-machine-learning-models.html/`. [Accessed: 17/09/2023].

Cretu, G. F., Stavrou, A., Locasto, M. E., Stolfo, S. J., and Keromytis, A. D. (2008). Casting out Demons: Sanitizing Training Data for Anomaly Sensors. In *IEEE Symposium on Security and Privacy*, pages 81–95.

Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al. (2004). Adversarial Classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108.

Daskalakis, C. and Panageas, I. (2018). The Limit Points of (Optimistic) Gradient Descent in Min-Max Optimization. In *Advances in Neural Information Processing Systems*, pages 9236–9246.

Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., and Roli, F. (2019). Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *USENIX security Symposium*, pages 321–338.

Dempe, S. (2002). *Foundations of Bilevel Programming*. Springer Science & Business Media.

Dempe, S. (2003). Annotated Bibliography on Bilevel Programming and Mathematical Programs with Equilibrium Constraints. *Optimization*, 52(3):333–359.

Dempe, S. (2018). *Bilevel optimization: theory, algorithms and applications*. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik.

Dempe, S., Kalashnikov, V., Pérez-Valdés, G. A., and Kalashnykova, N. (2015). *Bilevel Programming Problems: Theory, Algorithms and Applications to Energy Networks*. Springer.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., and Stewart, A. (2019). SEVER: A Robust Meta-Algorithm for Stochastic Optimization. In *International Conference on Machine Learning*, pages 1596–1606.

Doliwa, T., Fan, G., Simon, H. U., and Zilles, S. (2014). Recursive Teaching Dimension, VC-Dimension and Sample Compression. *Journal of Machine Learning Research*, 15(1):3107–3131.

Domke, J. (2012). Generic Methods for Optimization-Based Modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 318–326.

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Singh, A., and Poczos, B. (2017). Gradient Descent Can Take Exponential Time to Escape Saddle Points. In *Advances in Neural Information Processing Systems*, pages 1067–1077.

Durbin, J. (1954). Errors in Variables. *Revue de l'Institut International de Statistique/Review of the International Statistical Institute*, 22(1/3):23–32.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer.

Edmunds, T. A. and Bard, J. F. (1991). Algorithms for Nonlinear Bilevel Mathematical Programs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(1):83–89.

Ehlers, R., Grieser, J., Knieke, C., Rausch, A., and Schindler, M. (2019). Quality Assurance of Machine Learned Models by Integrating Domain Knowledge and Formal Verification. *Software Engineering and Software Management 2019*.

Ekins, S., Puhl, A. C., Zorn, K. M., Lane, T. R., Russo, D. P., Klein, J. J., Hickey, A. J., and Clark, A. M. (2019). Exploiting machine learning for end-to-end drug discovery and development. *Nature Materials*, 18(5):435–441.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust Physical-World Attacks on Deep Learning Visual Classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634.

Fang, M., Cao, X., Jia, J., and Gong, N. (2020). Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *USENIX Security Symposium*, pages 1605–1622.

Feng, J., Cai, Q.-Z., and Zhou, Z.-H. (2019). Learning to Confuse: Generating Training Time Adversarial Data with Auto-Encoder. In *Advances in Neural Information Processing Systems*, pages 11994–12004.

Feurer, M. and Hutter, F. (2019). Hyperparameter Optimization. *Automated Machine Learning: Methods, Systems, Challenges*, pages 3–33.

Fey, A., van der Hofstad, R., and Klok, M. J. (2008). Large deviations for eigenvalues of sample covariance matrices, with applications to mobile communication systems. *Advances in Applied Probability*, 40(4):1048–1071.

Figueiras-Vidal, A. R. (2013). *De Máquinas y Humanos. El Arte de la Toma de Decisiones*. Real Academia de Ingeniería, Madrid.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A Paradigm

for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395.

Floyd, S. and Warmuth, M. (1995). Sample Compression, Learnability, and the Vapnik-Chervonenkis Dimension. *Machine Learning*, 21:269–304.

Foo, C.-S., Do, C. B., and Ng, A. Y. (2008). Efficient multiple hyperparameter learning for log-linear models. In *Advances in Neural Information Processing Systems*, pages 377–384.

Fowl, L., Chiang, P.-y., Goldblum, M., Geiping, J., Bansal, A., Czaja, W., and Goldstein, T. (2021a). Preventing Unauthorized Use of Proprietary Data: Poisoning for Secure Dataset Release. In *ICLR Workshop on Security and Safety in Machine Learning Systems*.

Fowl, L. H., Goldblum, M., Chiang, P.-y., Geiping, J., Czaja, W., and Goldstein, T. (2021b). Adversarial Examples Make Strong Poisons. In *Advances in Neural Information Processing Systems*, pages 30339–30351.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. (2017). Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pages 1165–1173.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. (2018). Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *International Conference on Machine Learning*, pages 1568–1577.

Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333.

Frey, H. C. and Rhodes, D. S. (1996). Characterizing, simulating, and analyzing variability and uncertainty: an illustration of methods using an air toxics emissions example. *Human and Ecological Risk Assessment*, 2(4):762–797.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1.

Fu, S., He, F., Liu, Y., Shen, L., and Tao, D. (2022). Robust Unlearnable Examples:

Protecting Data Privacy Against Adversarial Learning. In *International Conference on Learning Representations*.

Fung, C., Yoon, C. J., and Beschastnikh, I. (2018). Mitigating Sybils in Federated Learning Poisoning. *arXiv preprint arXiv:1808.04866*.

Ge, R., Huang, F., Jin, C., and Yuan, Y. (2015). Escaping From Saddle Points—Online Stochastic Gradient for Tensor Decomposition. In *Conference on Learning Theory*, pages 797–842.

Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. (2021a). Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching. In *International Conference on Learning Representations*.

Geiping, J., Fowl, L., Somepalli, G., Goldblum, M., Moeller, M., and Goldstein, T. (2021b). What Doesn't Kill You Makes You Robust(er): Adversarial Training against Poisons and Backdoors. In *ICLR Workshop on Security and Safety in Machine Learning Systems*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*, volume 4. JHU Press.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Goodfellow, I., Shlens, J., and Szegedy, C. (2014b). Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. (2020). On the Iteration Complexity of Hypergradient Computation. In *International Conference on Machine Learning*, pages 3748–3758.

Griewank, A. and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, volume 105. SIAM.

Grosse, K., Bieringer, L., Besold, T. R., Biggio, B., and Krombholz, K. (2023). Machine Learning Security in Industry: A Quantitative Survey. *IEEE Transactions on Information Forensics and Security*, 18:1749–1762.

Gu, T., Liu, K., Dolan-Gavitt, B., and Garg, S. (2019). BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7:47230–47244.

Gull, S. F. (1989). Developments in Maximum Entropy Data Analysis. In *Maximum Entropy and Bayesian Methods*, pages 53–71. Springer.

Guttman, L. (1946). Enlargement Methods for Computing the Inverse Matrix. *The Annals of Mathematical Statistics*, pages 336–343.

Hammoudeh, Z. and Lowd, D. (2023). Reducing Certified Regression to Certified Classification for General Poisoning Attacks. In *IEEE Conference on Secure and Trustworthy Machine Learning*, pages 484–523.

Hampel, F. R. (1971). A General Qualitative Definition of Robustness. *The Annals of Mathematical Statistics*, 42(6):1887–1896.

Hansen, P., Jaumard, B., and Savard, G. (1992). New Branch-and-Bound Rules for Linear Bilevel Programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217.

Harrison Jr., D. and Rubinfeld, D. L. (1978). Hedonic Housing Prices and the Demand for Clean Air. *Journal of Environmental Economics and Management*, 5(1):81–102.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC press.

Hodge, V. and Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126.

Hodges, J. L. (1967). Efficiency in Normal Samples and Tolerance of Extreme Values for Some Estimates of Location. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 163–186.

Hong, S., Chandrasekaran, V., Kaya, Y., Dumitraş, T., and Papernot, N. (2020). On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping. *arXiv preprint arXiv:2002.11497*.

Houle, M. E. (2017). Local Intrinsic Dimensionality I: An Extreme-Value-Theoretic Foundation for Similarity Applications. In *International Conference on Similarity Search and Applications*, pages 64–79. Springer.

Houle, M. E., Schubert, E., and Zimek, A. (2018). On the Correlation Between Local Intrinsic Dimensionality and Outlierness. In *Similarity Search and Applications*, pages 177–191. Springer.

Huang, D., Cabral, R., and De la Torre, F. (2015). Robust Regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):363–375.

Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. (2021). Unlearnable Examples: Making Personal Data Unexploitable. In *International Conference on Learning Representations*.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I. P., and Tygar, J. D. (2011). Adversarial machine learning. In *ACM Workshop on Security and Artificial Intelligence*, pages 43–58.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. (2017a). Adversarial Attacks on Neural Network Policies. In *ICLR Workshop Track*.

Huang, W. R., Geiping, J., Fowl, L., Taylor, G., and Goldstein, T. (2020). MetaPoison: Practical General-purpose Clean-label Data Poisoning. In *Advances in Neural Information Processing Systems*, volume 33.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. (2017b). Safety Verification of Deep Neural Networks. In *International Conference on Computer Aided Verification*, pages 3–29.

Huber, P. J. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110:457–506.

Ilyas, A., Engstrom, L., and Madry, A. (2019). Prior Convictions: Black-Box Adversarial Attacks with Bandits and Priors. In *International Conference on Learning Representations*.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018). Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *IEEE Symposium on Security and Privacy*, pages 19–35.

Jagielski, M., Severi, G., Pousette Harger, N., and Oprea, A. (2021). Subpopulation Data Poisoning Attacks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 3104–3122.

Jia, J., Cao, X., and Gong, N. Z. (2021). Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks. In *AAAI Conference on Artificial Intelligence*, pages 7961–7969.

Jia, J., Liu, Y., Cao, X., and Gong, N. Z. (2022). Certified Robustness of Nearest Neighbors against Data Poisoning and Backdoor Attacks . In *AAAI Conference on Artificial Intelligence*, pages 9575–9583.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. (2017). How to Escape Saddle Points Efficiently. In *International Conference on Machine Learning*, pages 1724–1732.

Jylänki, P., Vanhatalo, J., and Vehtari, A. (2011). Robust Gaussian Process Regression with a Student-t Likelihood. *Journal of Machine Learning Research*, 12(11).

Kalashnikov, V. V., Dempe, S., Pérez-Valdés, G. A., Kalashnykova, N. I., and Camacho-Vallejo, J.-F. (2015). Bilevel Programming and Applications. *Mathematical Problems in Engineering*, 2015.

Kan, W. (2013). Lending Club Loan Data: Analyze Lending Club's Issued Loans. [Online]. Available from: `https://www.kaggle.com/wendykan/lending-club-loan-data`.

Karush, W. (1939). Minima of Functions of Several Variables with Inequalities as Side Conditions. Master's thesis, Dept. of Mathematics, Univ. of Chicago.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *International Conference on Computer Aided Verification*, pages 97–117.

Kendall, A. (2017). Deep Learning Is Not Good Enough, We Need Bayesian Deep Learning for Safe AI. [Online]. Available from: `https://alexgkendall.com/computer_vision/bayesian_deep_learning_for_safe_ai/`. [Accessed: 27/09/2023].

Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590.

Kerckhoffs, A. (1883). La Cryptographie Militaire. *Journal des Sciences Militaires*, pages 5–38.

Kioustelidis, J. B. (1986). Bounds for positive roots of polynomials. *Journal of Computational and Applied Mathematics*, 16(2):241–244.

Kloft, M. and Laskov, P. (2012). Security Analysis of Online Centroid Anomaly Detection. *Journal of Machine Learning Research*, 13(Dec):3681–3724.

Koh, P. W. and Liang, P. (2017). Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*, pages 1885–1894.

Koh, P. W., Steinhardt, J., and Liang, P. (2022). Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, pages 1–47.

Kolstad, C. D. and Lasdon, L. S. (1990). Derivative Evaluation and Computational Experience with Large Bilevel Mathematical Programs. *Journal of Optimization Theory and Applications*, 65(3):485–499.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtarik, P., Suresh, A. T., and Bacon, D. (2016). Federated Learning: Strategies for Improving Communication Efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*.

Kos, J. and Song, D. (2017). Delving into Adversarial Attacks on Deep Policies. In *ICLR Workshop Track*.

Krizhevsky, A. et al. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear Programming. In *Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492.

Kumar, A., Zhang, H., Singla, A., and Chen, Y. (2021). The Teaching Dimension of Kernel Perceptron. In *International Conference on Artificial Intelligence and Statistics*, pages 2071–2079.

Kumar, R. S. S., Nyström, M., Lambert, J., Marshall, A., Goertzel, M., Comissoneru, A., Swann, M., and Xia, S. (2020). Adversarial Machine Learning-Industry Perspectives. In *IEEE Security and Privacy Workshops*, pages 69–75.

Kuncheva, L. I. (2007). A Stability Index for Feature Selection. In *IASTED International Multi-Conference: Artificial Intelligence and Applications*, pages 390–395.

Kuss, M. (2006). *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, Technische Universität Darmstadt.

Lai, Y.-J. (1996). Hierarchical optimization: a satisfactory solution. *Fuzzy Sets and Systems*, 77(3):321–335.

Larsen, J., Hansen, L. K., Svarer, C., and Ohlsson, M. (1996). Design and Regularization of Neural Networks: The Optimal Use of a Validation Set. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 62–71.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018). Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016). Gradient Descent Only Converges to Minimizers. In *Conference on Learning Theory*, pages 1246–1257.

Levine, A. and Feizi, S. (2021). Deep Partition Aggregation: Provable Defense against General Poisoning Attacks. In *International Conference on Learning Representations*.

Levy, K. Y. (2016). The Power of Normalization: Faster Evasion of Saddle Points. *arXiv preprint arXiv:1611.04831*.

Li, C. and Qiu, M. (2019). *Reinforcement Learning for Cyber-Physical Systems with Cybersecurity Case Studies*. CRC Press.

Li, F., Lai, L., and Cui, S. (2021). Optimal Feature Manipulation Attacks Against Linear Regression. *IEEE Transactions on Signal Processing*, 69:5580–5594.

Li, P., Li, J., Huang, Z., Gao, C.-Z., Chen, W.-B., and Chen, K. (2018). Privacy-preserving outsourced classification in cloud computing. *Cluster Computing*, 21(1):277–286.

Li, R.-C. (2006). Matrix Perturbation Theory. In *Handbook of Linear Algebra*, pages 15–1–15–18. Chapman and Hall/CRC.

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22.

Lim, S. H., Xu, H., and Mannor, S. (2013). Reinforcement Learning in Robust Markov Decision Processes. In *Neural Information Processing Systems*, pages 701–709.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. (2017). Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *International Joint Conference on Artificial Intelligence*, pages 3756–3762.

Liu, C., Li, B., Vorobeychik, Y., and Oprea, A. (2017). Robust Linear Regression against Training Data Poisoning. In *ACM Workshop on Artificial Intelligence and Security*, pages 91–102.

Liu, H., Simonyan, K., and Yang, Y. (2019). DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Liu, J., Zhu, X., and Ohannessian, H. (2016). The Teaching Dimension of Linear Learners. In *International Conference on Machine Learning*, pages 117–126.

Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S., and Leung, V. C. (2018a). A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View. *IEEE Access*, 6:12103–12117.

Liu, T. Y., Yang, Y., and Mirzasoleiman, B. (2022). Friendly Noise against Adversarial Noise: A Powerful Defense against Data Poisoning Attack. *Advances in Neural Information Processing Systems*, pages 11947–11959.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. (2018b). Trojaning Attack on Neural Networks. In *The Network and Distributed System Security Symposium*.

Loshchilov, I. and Glasmachers, T. (2019). Black Box Optimization Competition. [Online]. Available from: `https://www.ini.rub.de/PEOPLE/glasmtbl/projects/bbcomp/`. [Accessed: 12/09/2023].

Lowd, D. and Meek, C. (2005a). Adversarial learning. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 641–647.

Lowd, D. and Meek, C. (2005b). Good Word Attacks on Statistical Spam Filters. In *Conference on Email and Anti-Spam*.

Lu, Y., Kamath, G., and Yu, Y. (2023). Exploring the Limits of Model-Targeted Indiscriminate Data Poisoning Attacks. In *International Conference on Machine Learning*.

Ma, Y., Zhu, X., and Hsu, J. (2019). Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *International Joint Conference on Artificial Intelligence*, pages 4732–4738.

MacKay, D. J. C. (1992). Bayesian Interpolation. *Neural Computation*, 4(3):415–447.

Mackay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. (2019). Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. In *International Conference on Learning Representations*.

Maclaurin, D., Duvenaud, D., and Adams, R. (2015). Gradient-based Hyperparameter Optimization through Reversible Learning. In *International Conference on Machine Learning*, pages 2113–2122.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395.

McDaniel, P., Papernot, N., and Celik, Z. B. (2016). Machine Learning in Adversarial Settings. *IEEE Security & Privacy*, 14(3):68–72.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics*, pages 1273–1282.

Mehra, A., Kailkhura, B., Chen, P.-Y., and Hamm, J. (2021). How Robust Are Randomized Smoothing Based Defenses to Data Poisoning? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13244–13253.

Mehrabi, N., Naveed, M., Morstatter, F., and Galstyan, A. (2021). Exacerbating Algorithmic Bias through Fairness Attacks. In *AAAI Conference on Artificial Intelligence*, pages 8930–8938.

Mei, S. and Zhu, X. (2015). Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *AAAI Conference on Artificial Intelligence*, pages 2871–2877.

Meligkotsidou, L., Vrontos, I. D., and Vrontos, S. D. (2009). Quantile regression analysis of hedge fund strategies. *Journal of Empirical Finance*, 16(2):264–279.

Mercer, J. (1909). Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London Series A*, 209:415–446.

Meseguer, P., Rossi, F., and Schiex, T. (2006). Soft Constraints. In *Foundations of Artificial Intelligence*, volume 2, pages 281–328. Elsevier.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving Deep Neural Networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier.

Milojković, N., Antognini, D., Bergamin, G., Faltings, B., and Musat, C. (2019). Multi-Gradient Descent for Multi-Objective Recommender Systems. *arXiv preprint arXiv:2001.00846*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2574–2582.

Moreau, J. J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299.

Müller, N., Kowatsch, D., and Böttinger, K. (2020). Data Poisoning Attacks on Regression Learning and Corresponding Defenses. In *IEEE Pacific Rim International Symposium on Dependable Computing*, pages 80–89.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. (2017). Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In *ACM Workshop on Artificial Intelligence and Security*, pages 27–38.

Muñoz-González, L., Carnerero-Cano, J., Co, K. T., and Lupu, E. C. (2019a). Challenges and Advances in Adversarial Machine Learning. In *Resilience and Hybrid Threats: Security and Integrity for the Digital World*, volume 55, page 102. IOS Press.

Muñoz-González, L. and Lupu, E. C. (2018). The Secret of Machine Learning. *ITNow*, 60(1):38–39.

Muñoz-González, L. and Lupu, E. C. (2019). The Security of Machine Learning Systems. In *AI in Cybersecurity*, pages 47–79. Springer.

Muñoz-González, L., Pfitzner, B., Russo, M., Carnerero-Cano, J., and Lupu, E. C. (2019b). Poisoning Attacks with Generative Adversarial Nets. *arXiv preprint arXiv:1906.07773*.

Murray, R., Swenson, B., and Kar, S. (2019). Revisiting Normalized Gradient Descent: Fast Evasion of Saddle Points. *IEEE Transactions on Automatic Control*, 64(11):4818–4824.

Neff, G. and Nagy, P. (2016). Talking to Bots: Symbiotic Agency and the Case of Tay. *International Journal of Communication*, 10:4915–4931.

Nelson, B., Barreno, M., Chi, F. J., Joseph, A. D., Rubinstein, B. I. P., Saini, U., Sutton, C. A., Tygar, J. D., and Xia, K. (2008). Exploiting Machine Learning to Subvert Your Spam Filter. In *USENIX Workshop on Large Scale Exploits and Emergent Threats*.

Nesterov, Y. (2013). *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media.

Newey, W. K. and West, K. D. (1987). A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, pages 703–708.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 427–436.

Osborne, M. (2010). Asymptotic behaviour in linear least squares problems. *IMA Journal of Numerical Analysis*, 30(1):241–247.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744.

Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Papernot, N. (2018). A Marauder's Map of Security and Privacy in Machine Learning: An overview of current and future research directions for making machine learning secure and private. In *ACM Workshop on Artificial Intelligence and Security*, pages 1–1.

Papernot, N., McDaniel, P., and Goodfellow, I. (2016a). Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Sample. *arXiv preprint arXiv:1605.07277*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical Black-Box Attacks against Machine Learning. In *ACM Asia Conference on Computer and Communications Security*, pages 506–519.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016b). The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy*, pages 372–387.

Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. P. (2018). SoK: Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy*, pages 399–414.

Paudice, A., Muñoz-González, L., György, A., and Lupu, E. C. (2018a). Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection. *arXiv preprint arXiv:1802.03041*.

Paudice, A., Muñoz-González, L., and Lupu, E. C. (2018b). Label Sanitization Against Label Flipping Poisoning Attacks. In *ECML PKDD Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018*, pages 5–15.

Paulson, D. S. (2006). *Handbook of Regression and Modeling: Applications for the Clinical and Pharmaceutical Industries*. CRC Press.

Pearlmutter, B. A. (1994). Fast Exact Multiplication by the Hessian. *Neural Computation*, 6(1):147–160.

Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, pages 737–746.

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., and Cavallaro, L. (2019). TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *USENIX Security Symposium*, pages 729–746.

Pulina, L. and Tacchella, A. (2010). An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *International Conference on Computer Aided Verification*, pages 243–257.

Qazaz, C. S., Williams, C. K. I., and Bishop, C. M. (1997). An Upper Bound on the Bayesian Error Bars for Generalized Linear Regression. In *International Conference on Mathematics of Neural Networks: Models, Algorithms and Applications*, pages 295–299.

Qian, H., Liu, X.-H., Su, C.-X., Zhou, A., and Yu, Y. (2022). The Teaching Dimension of Regularized Kernel Learners. In *International Conference on Machine Learning*, pages 17984–18002.

Rakhsha, A., Radanovic, G., Devidze, R., Zhu, X., and Singla, A. (2021). Policy Teaching in Reinforcement Learning via Environment Poisoning Attacks. *Journal of Machine Learning Research*, 22(1):9567–9611.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. In *International Conference on Machine Learning*, pages 8821–8831.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*, volume 1. MIT Press.

Rezaei, K., Banihashem, K., Chegini, A., and Feizi, S. (2023). Run-Off Election: Improved Provable Defense against Data Poisoning Attacks. In *International Conference on Machine Learning*.

Rockafellar, R. T. (1966). Characterization of the subdifferentials of convex functions. *Pacific Journal of Mathematics*, 17(3):497–510.

Rockafellar, R. T. (1980). Generalized Directional Derivatives and Subgradients of Nonconvex Functions. *Canadian Journal of Mathematics*, 32(2):257–280.

Rosen, K. H. (1999). *Handbook of Discrete and Combinatorial Mathematics*. CRC press.

Rosenfeld, E., Winston, E., Ravikumar, P., and Kolter, Z. (2020). Certified Robustness to Label-Flipping Attacks via Randomized Smoothing. In *International Conference on Machine Learning*, pages 8230–8241.

Rousseeuw, P. J. (1984). Least Median of Squares Regression. *Journal of the American Statistical Association*, 79(388):871–880.

Rubinstein, B. I. P., Nelson, B., Huang, L., Joseph, A. D., Lau, S.-h., Rao, S., Taft, N., and Tygar, J. D. (2009). ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors. In *ACM SIGCOMM Conference on Internet Measurement*, pages 1–14.

Rybski, P. E., Yoon, K., Stolarz, J., and Veloso, M. M. (2007). Interactive Robot Task Training through Dialog and Demonstration. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 49–56.

Salem, O., Guerassimov, A., Mehaoua, A., Marcus, A., and Furht, B. (2014). Anomaly Detection in Medical Wireless Sensor Networks using SVM and Linear Regression Models. *International Journal of E-Health and Medical Communications*, 5(1):20–45.

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229.

Savard, G. and Gauvin, J. (1994). The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15(5):265–272.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018). Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175.

Shan, S., Wenger, E., Wang, B., Li, B., Zheng, H., and Zhao, B. Y. (2020). Gotta catch 'em all: Using honeypots to catch adversarial attacks on neural networks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 67–83.

Shannon, C. E. (1949). Communication Theory of Secrecy Systems. *The Bell System Technical Journal*, 28(4):656–715.

Shen, J., Zhu, X., and Ma, D. (2019). TensorClog: An Imperceptible Poisoning Atack on Deep Neural Network Applications. *IEEE Access*, 7:41498–41506.

Sherman, J. and Morrison, W. J. (1950). Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics*, 21(1):124–127.

Shewchuk, J. R. (1994). An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report, Carnegie Mellon University.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *International Conference on Document Analysis and Recognition*, pages 958–962.

Sinha, A., Malo, P., and Deb, K. (2013). Efficient Evolutionary Algorithm for Single-Objective Bilevel Optimization. *arXiv preprint arXiv:1303.3901*.

Sinha, A., Malo, P., and Deb, K. (2018). A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

Solans, D., Biggio, B., and Castillo, C. (2020). Poisoning Attacks on Algorithmic Fairness. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 162–177. Springer.

Steinhardt, J., Koh, P. W. W., and Liang, P. S. (2017). Certified Defenses for Data Poisoning Attacks. In *Advances in Neural Information Processing Systems*, pages 3520–3532.

Sturm, J. C. F. (1829). Mémoire sur la résolution des équations numériques. *Bulletin des Sciences de Férussac*, 11:419–425.

Suciu, O., Marginean, R., Kaya, Y., Daume III, H., and Dumitras, T. (2018). When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *USENIX Security Symposium*, pages 1299–1316.

Sun, C. and Hou, J. (2017). An improved principal component regression for quality-related process monitoring of industrial control systems. *IEEE Access*, 5:21723–21730.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tao, L., Feng, L., Yi, J., Huang, S.-J., and Chen, S. (2021). Better Safe Than Sorry: Preventing Delusive Adversaries with Adversarial Training. In *Advances in Neural Information Processing Systems*, pages 16209–16225.

Tipping, M. E. (2001). Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244.

Troy, C. (2016). Machine Learning, Part 2: Classification . [Online]. Available from: `https://chelseatroy.com/2016/03/22/machine-learning-part-2-classification/`. [Accessed: 14/09/2023].

Van, M.-H., Du, W., Wu, X., and Lu, A. (2022). Poisoning Attacks on Fair Machine Learning. In *International Conference on Database Systems for Advanced Applications*, pages 370–386. Springer.

Vicente, L., Savard, G., and Júdice, J. (1994). Descent Approaches for Quadratic Bilevel Programming. *Journal of Optimization Theory and Applications*, 81(2):379–399.

Vicente, L. N. and Calamai, P. H. (1994). Bilevel and Multilevel Programming: A Bibliography Review. *Journal of Global optimization*, 5(3):291–306.

von Neumann, J. (1928). Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320.

von Stackelberg, H. (1934). *Marktform und Gleichgewicht*. Springer.

Wang, B., Cao, X., Gong, N. Z., et al. (2020). On Certifying Robustness against Backdoor Attacks via Randomized Smoothing. In *CVPR Workshop on Adversarial Machine Learning in Computer Vision*.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. (2019). Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *IEEE Symposium on Security and Privacy*, pages 707–723.

Wang, W., Levine, A. J., and Feizi, S. (2022). Improved Certified Defenses against Data Poisoning with (Deterministic) Finite Aggregation. In *International Conference on Machine Learning*, pages 22769–22783.

Wang, Y. and Chaudhuri, K. (2018). Data Poisoning Attacks against Online Learning. *arXiv preprint arXiv:1808.08994*.

Weerasinghe, S., Abraham, T., Alpcan, T., Erfani, S. M., Leckie, C., and Rubinstein, B. I. (2022). Local Intrinsic Dimensionality Signals Adversarial Perturbations. In *IEEE Conference on Decision and Control*, pages 6118–6125.

Weerasinghe, S., Abraham, T., Alpcan, T., Erfani, S. M., Leckie, C., and Rubinstein, B. I. P. (2021). Closing the BIG-LID: An Effective Local Intrinsic Dimensionality Defense for Nonlinear Regression Poisoning. In *International Joint Conference on Artificial Intelligence*, pages 3176–3184.

Wei, M. (1990). Perturbation of the Least Squares Problem. *Linear Algebra and Its Applications*, 141:177–182.

Wein, L. M. (2009). OR forum—Homeland Security: From Mathematical Models to Policy Implementation: The 2008 Philip McCord Morse Lecture. *Operations Research*, 57(4):801–811.

Wen, J., Zhao, B. Z. H., Xue, M., Oprea, A., and Qian, H. (2021). With Great Dispersion Comes Greater Resilience: Efficient Poisoning Attacks and Defenses for Linear Regression Models. *IEEE Transactions on Information Forensics and Security*.

Wen, Y., Rahman, M. F., Xu, H., and Tseng, T.-L. B. (2022). Recent advances and

trends of predictive maintenance from data-driven machine prognostics perspective. *Measurement*, 187:110276.

Weyl, H. (1912). Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479.

Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., and Roli, F. (2015). Is Feature Selection Secure against Training Data Poisoning? In *International Conference on Machine Learning*, pages 1689–1698.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, H., Caramanis, C., and Mannor, S. (2011). Sparse Algorithms Are Not Stable: A No-Free-Lunch Theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):187–193.

Xu, H., Caramanis, C., and Mannor, S. (2012). Outlier-Robust PCA: The High-Dimensional Case. *IEEE Transactions on Information Theory*, 59(1):546–572.

Yang, C., Wu, Q., Li, H., and Chen, Y. (2017). Generative Poisoning Attack Method Against Neural Networks. *arXiv preprint arXiv:1703.01340*.

Yang, Y., Liu, T. Y., and Mirzasoleiman, B. (2022). Not All Poisons are Created Equal: Robust Training against Data Poisoning. In *International Conference on Machine Learning*, pages 25154–25165.

Yao, Y., Viswanath, B., Cryan, J., Zheng, H., and Zhao, B. Y. (2017). Automated Crowdturfing Attacks and Defenses in Online Review Systems. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1143–1158.

Yuan, C.-H. and Wu, S.-H. (2021). Neural Tangent Generalization Attacks. In *International Conference on Machine Learning*, pages 12230–12240.

Zhang, X., Zhu, X., and Lessard, L. (2020). Online Data Poisoning Attacks. In *Learning for Dynamics and Control*, pages 201–210.

Zhang, X., Zhu, X., and Wright, S. (2018). Training Set Debugging Using Trusted Items. In *AAAI Conference on Artificial Intelligence*, pages 4482–4489.

Zhao, S.-Y., Xie, Y.-P., and Li, W.-J. (2021). On the convergence and improvement of stochastic normalized gradient descent. *Science China Information Sciences*, 64:1–13.

Zhu, C., Huang, W. R., Li, H., Taylor, G., Studer, C., and Goldstein, T. (2019). Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *International Conference on Machine Learning*, pages 7614–7623.

Zhu, X. (2013). Machine Teaching for Bayesian Learners in the Exponential Family. In *Advances in Neural Information Processing Systems*, pages 1905–1913.

Zhu, X., Singla, A., Zilles, S., and Rafferty, A. N. (2018). An Overview of Machine Teaching. *arXiv preprint arXiv:1801.05927*.

Zoph, B. and Le, Q. (2017). Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*.

Appendices

APPENDIX A

Woodbury Formula, Sherman-Morrison Formula, and Matrix Determinant Lemma

A.1 Woodbury Formula

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{U} \in \mathbb{R}^{n \times k}$, $\mathbf{C} \in \mathbb{R}^{k \times k}$, and $\mathbf{V} \in \mathbb{R}^{k \times n}$ be conformable matrices, where $\mathbf{A}$ and $\mathbf{C}$ are non-singular. The Woodbury matrix formula says that the inverse of a rank-k correction of some matrix can be computed by doing a rank-k correction to the inverse of the original matrix. The Woodbury formula (Guttman, 1946) is:

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1}, \tag{A.1}$$

where the Schur complement of $\mathbf{A}$: $\mathbf{A}_S = \mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}$ is also assumed to be non-singular. This result can be derived using blockwise matrix inversion.

A.2 Sherman-Morrison Formula and Matrix Determinant Lemma

The Sherman-Morrison formula (Sherman and Morrison, 1950) computes the inverse of the sum of a non-singular matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and the outer product, $\mathbf{u}\mathbf{v}^\mathsf{T}$, of column vectors $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^n$. In other words, it computes the inverse of a rank-1 correction of the matrix $\mathbf{A}$. The Sherman-Morrison formula—special case of the Woodbury formula—is:

$$\left(\mathbf{A} + \mathbf{u}\mathbf{v}^\mathsf{T}\right)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^\mathsf{T}\mathbf{A}^{-1}}{1 + \mathbf{v}^\mathsf{T}\mathbf{A}^{-1}\mathbf{u}}. \tag{A.2}$$

This result can be obtained using blockwise matrix inversion. On the other hand, assuming $\mathbf{A}$ is non-singular, the matrix determinant lemma states that:

$$\det\left(\mathbf{A} + \mathbf{u}\mathbf{v}^\mathsf{T}\right) = \left(1 + \mathbf{v}^\mathsf{T}\mathbf{A}^{-1}\mathbf{u}\right) \det\left(\mathbf{A}\right) .PN \tag{A.3}$$

Thus, $\mathbf{A} + \mathbf{u}\mathbf{v}^\mathsf{T}$ is non-singular if and only if $1 + \mathbf{v}^\mathsf{T}\mathbf{A}^{-1}\mathbf{u} \neq 0$. In the particular case where $\mathbf{A}$ is positive definite and $\mathbf{u} = \mathbf{v}$, $\mathbf{A} + \mathbf{u}\mathbf{u}^\mathsf{T}$ is always non-singular because $\mathbf{u}^\mathsf{T}\mathbf{A}^{-1}\mathbf{u} > 0$.

If $\mathbf{A}^{-1}$ is already known, the formula provides a numerically cheap way to compute the inverse of $\mathbf{A}$ perturbed by the matrix $\mathbf{u}\mathbf{v}^\mathsf{T}$. The computation is relatively cheap because $(\mathbf{A} + \mathbf{u}\mathbf{v}^\mathsf{T})^{-1}$ does not have to be computed from scratch (which in general is expensive), but can be computed by perturbing $\mathbf{A}^{-1}$.

APPENDIX B

Inverse of Block Matrix Whose Diagonal Blocks Are Non-Singular

Let $\mathbf{P} \in \mathbb{R}^{(n+m)\times(n+m)}$ be a non-singular matrix partitioned into four blocks:

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \tag{B.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{m\times n}$, and $\mathbf{D} \in \mathbb{R}^{m\times m}$. If both $\mathbf{A}$ and $\mathbf{D}$ non-singular, $\mathbf{P}$ can be inverted blockwise as follows:

$$\mathbf{P}^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{0}_{n\times m} \\ \mathbf{0}_{m\times n} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I}_n & -\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I}_m \end{bmatrix}. \tag{B.2}$$

This result can be derived by performing a block Gaussian elimination on the matrix $\mathbf{P}$.

APPENDIX C

Bounds on Bilinear Quadratic Forms

Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a real symmetric matrix. Since $\mathbf{A}$ is real and symmetric, it is orthogonally diagonalisable. Let $\lambda_i = \lambda_i(\mathbf{A}),\ i = 1, \ldots, m$, and $\mathbf{q}_i = \mathbf{q}_i(\mathbf{A}),\ i = 1, \ldots, m$, denote the eigenvalues and orthonormal eigenvectors of $\mathbf{A}$, respectively. Let $\lambda_{\min} = \min_i \lambda_i(\mathbf{A})$ and $\lambda_{\max} = \max_i \lambda_i(\mathbf{A})$ be the smallest and largest eigenvalues of $\mathbf{A}$, respectively, and let $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ be their corresponding orthonormal eigenvectors. We shall show that, for a vector $\mathbf{x} \in \mathbb{R}^m$, the bilinear quadratic form $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}$ can be bounded as:

$$\lambda_{\min}||\mathbf{x}||_2^2 \leq \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} \leq \lambda_{\max}||\mathbf{x}||_2^2. \tag{C.1}$$

Because $\mathbf{A}$ is orthogonally diagonalisable: $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathsf{T}$, where $\mathbf{Q} \in \mathbb{R}^{m \times m}$ is an orthogonal matrix whose ith column is $\mathbf{q}_i$, and $\mathbf{\Lambda} \in \mathbb{R}^{m \times m}$ is a diagonal matrix whose ith diagonal entry is λ_i. Then:

$$\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathsf{T}\mathbf{x} = \left(\mathbf{Q}^\mathsf{T}\mathbf{x}\right)^\mathsf{T} \mathbf{\Lambda} \left(\mathbf{Q}^\mathsf{T}\mathbf{x}\right) = \sum_{i=1}^{m} \lambda_i \left(\mathbf{q}_i^\mathsf{T}\mathbf{x}\right)^2. \tag{C.2}$$

We can bound the previous expression as:

$$\sum_{i=1}^{m} \lambda_{\min} \left(\mathbf{q}_i^\mathsf{T}\mathbf{x}\right)^2 \leq \sum_{i=1}^{m} \lambda_i \left(\mathbf{q}_i^\mathsf{T}\mathbf{x}\right)^2 \leq \sum_{i=1}^{m} \lambda_{\max} \left(\mathbf{q}_i^\mathsf{T}\mathbf{x}\right)^2. \tag{C.3}$$

Because $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^\mathsf{T} = \mathbf{I}_m$, where $\mathbf{I}_m$ is the identity matrix, we can write:

$$\sum_{i=1}^{m} \left(\mathbf{q}_i^\mathsf{T}\mathbf{x}\right)^2 = \left|\left|\mathbf{Q}^\mathsf{T}\mathbf{x}\right|\right|_2^2 = \left(\mathbf{Q}^\mathsf{T}\mathbf{x}\right)^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{Q}^\mathsf{T}\mathbf{x} = ||\mathbf{x}||_2^2. \tag{C.4}$$

Therefore:

$$\lambda_{\min}||\mathbf{x}||_2^2 \leq \sum_{i=1}^{m} \lambda_i \left(\mathbf{q}_i^\mathsf{T}\mathbf{x}\right)^2 \leq \lambda_{\max}||\mathbf{x}||_2^2, \tag{C.5}$$

so that:

$$\lambda_{\min}||\mathbf{x}||_2^2 \leq \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} \leq \lambda_{\max}||\mathbf{x}||_2^2, \tag{C.6}$$

as required. Note also that $\mathbf{q}_{\min}^\mathsf{T}\mathbf{A}\mathbf{q}_{\min} = \lambda_{\min}||\mathbf{q}_{\min}||_2^2 = \lambda_{\min}$ and $\mathbf{q}_{\max}^\mathsf{T}\mathbf{A}\mathbf{q}_{\max} = \lambda_{\max}||\mathbf{q}_{\max}||_2^2 = \lambda_{\max}$, so the bounds are tight.

APPENDIX D

Robustness of Update of Positive Definite and Positive Semidefinite Matrices

Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a symmetric and positive semidefinite matrix—which can also be positive definite—and let $\mathbf{B} \in \mathbb{R}^{m \times m}$ be some other symmetric and positive semidefinite matrix—which, again, can also be positive definite. Consider the perturbed matrix $\mathbf{C} = \mathbf{A} + \varepsilon\mathbf{B}$, for some $\varepsilon \in \mathbb{R}_{>0}$. We shall analyse the conditions under which $\mathbf{C}$ is positive definite or positive semidefinite (singular).

Let us define $\mathbf{x} \in \mathbb{R}^m$. Then: $\mathbf{x}^\mathsf{T}\mathbf{C}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} + \varepsilon\mathbf{x}^\mathsf{T}\mathbf{B}\mathbf{x}$. Moreover, $\lambda_{\min}(\mathbf{A})||\mathbf{x}||_2^2 \leq \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} \leq \lambda_{\max}(\mathbf{A})||\mathbf{x}||_2^2$ and $\varepsilon\lambda_{\min}(\mathbf{B})||\mathbf{x}||_2^2 \leq \varepsilon\mathbf{x}^\mathsf{T}\mathbf{B}\mathbf{x} \leq \varepsilon\lambda_{\max}(\mathbf{B})||\mathbf{x}||_2^2$, where $\lambda_{\min}(\,\cdot\,)$ and $\lambda_{\max}(\,\cdot\,)$ denote the minimum and maximum eigenvalues of the corresponding matrices (see Appx. C). Then:

$$(\lambda_{\min}(\mathbf{A}) + \varepsilon\lambda_{\min}(\mathbf{B}))\,||\mathbf{x}||_2^2 \leq \mathbf{x}^\mathsf{T}\mathbf{C}\mathbf{x} \leq (\lambda_{\max}(\mathbf{A}) + \varepsilon\lambda_{\max}(\mathbf{B}))\,||\mathbf{x}||_2^2. \tag{D.1}$$

By definition, we know that $\mathbf{C}$ is positive definite if and only $\mathbf{x}^\mathsf{T}\mathbf{C}\mathbf{x} > 0$, that is, $\lambda_{\min}(\mathbf{A}) + \varepsilon\lambda_{\min}(\mathbf{B}) > 0$. Thus, if $\mathbf{A}$ is positive definite and $\mathbf{B}$ is positive definite or positive semidefinite, then $\mathbf{C}$ is always positive definite, because $\lambda_{\min}(\mathbf{A}) > 0$ and $\lambda_{\min}(\mathbf{B}) \geq 0$. If $\mathbf{A}$ is positive semidefinite and $\mathbf{B}$ is positive definite, then $\mathbf{C}$ is always positive definite

as well, because $\lambda_{\min}(\mathbf{A}) \geq 0$ and $\lambda_{\min}(\mathbf{B}) > 0$. However, if both $\mathbf{A}$ and $\mathbf{B}$ are positive semidefinite, more information is needed to determine if $\mathbf{C}$ is positive definite, since in this case $\mathbf{x}^\mathsf{T}\mathbf{C}\mathbf{x} \geq 0$, but it could be strictly positive.

For instance, let us assume $\mathbf{A}$ and $\mathbf{B}$ are positive semidefinite and $\mathbf{B} = \mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T}$, where $\mathbf{I}_m$ is the identity matrix, $\mathbf{e}_i^\mathsf{T}$ is a basis vector: $\mathbf{e}_i = \left[\mathbf{0}_{i-1}^\mathsf{T}\ 1\ \mathbf{0}_{m-i}^\mathsf{T}\right]^\mathsf{T}, i \in \{1, \ldots, n\}$, and $\mathbf{0}_{i-1}$ and $\mathbf{0}_{m-i}$ are column vectors of zeros. $\mathbf{B}$ is positive semidefinite because: $\mathbf{B} = \mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T} = \left(\mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right)^\mathsf{T}\left(\mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right) = \left(\mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right)\left(\mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right)^\mathsf{T}$ and $\mathbf{e}_i^\mathsf{T}\left(\mathbf{I}_m - \mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right)\mathbf{e}_i = 0$.

In ML, this kind of matrices can occur, for example, in the scatter matrix in ridge regression models, when one the model's parameters is not regularised. In this case, ε corresponds to the regularisation hyperparameter.

Thus, $\mathbf{C} = (\mathbf{A} + \varepsilon\mathbf{I}_m) - \varepsilon\mathbf{e}_i\mathbf{e}_i^\mathsf{T}$, where $\mathbf{A} + \varepsilon\mathbf{I}_m$ is positive definite. Because $\mathbf{C}$ is a rank-1 update of $\mathbf{A} + \varepsilon\mathbf{I}_m$, we can use the matrix determinant lemma (Appx. A.2) to compute:

$$\det\left((\mathbf{A} + \varepsilon\mathbf{I}_m) - \varepsilon\mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right) = \left(1 - \mathbf{e}_i^\mathsf{T}\left(\varepsilon^{-1}\mathbf{A} + \mathbf{I}_m\right)^{-1}\mathbf{e}_i\right)\det\left(\mathbf{A} + \varepsilon\mathbf{I}_m\right). \tag{D.2}$$

Hence, $\mathbf{C}$ is positive semidefinite (singular) if and only if $\mathbf{e}_i^\mathsf{T}\left(\varepsilon^{-1}\mathbf{A} + \mathbf{I}_m\right)^{-1}\mathbf{e}_i = 1$, i.e., if and only if $\left[\left(\varepsilon^{-1}\mathbf{A} + \mathbf{I}_m\right)^{-1}\right]_{i,i} = 1$, which corresponds to the ith entry of the diagonal of $\left(\varepsilon^{-1}\mathbf{A} + \mathbf{I}_m\right)^{-1}$. Therefore, $\mathbf{C}$ is positive semidefinite only when:

$$\lim_{\varepsilon \to \infty}\left(\varepsilon^{-1}\mathbf{A} + \mathbf{I}_m\right)^{-1} = \mathbf{I}_m. \tag{D.3}$$

In ML settings, this situation can happen when the regularisation hyperparameter is too large, which leads to underfitting.

In summary, if at least $\mathbf{A}$ or $\mathbf{B}$ are positive definite, then $\mathbf{C}$ is also positive definite. If both $\mathbf{A}$ and $\mathbf{B}$ are positive semidefinite, then more information is required to determine whether $\mathbf{C}$ is positive definite.

APPENDIX E

On the Rank of Matrices

E.1 Rank of Scatter and Gram Matrices

Here we shall show that, for a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$:

$$\operatorname{rank}(\mathbf{A}) = \operatorname{rank}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right). \tag{E.1}$$

Since the scatter matrix of $\mathbf{A}$,[1] $\mathbf{A}^\mathsf{T}\mathbf{A} \in \mathbb{R}^{m \times m}$, has the same number of columns as $\mathbf{A}$, using the rank-nullity theorem:

$$\operatorname{rank}(\mathbf{A}) + \operatorname{nullity}(\mathbf{A}) = m = \operatorname{rank}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right) + \operatorname{nullity}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right), \tag{E.2}$$

where $\operatorname{rank}(\mathbf{A}) = \dim(\operatorname{col}(\mathbf{A})) = \dim(\operatorname{row}(\mathbf{A}))$, $\operatorname{nullity}(\mathbf{A}) = \dim(\ker(\mathbf{A}))$, and $\operatorname{col}(\mathbf{A})$, $\operatorname{row}(\mathbf{A})$, and $\ker(\mathbf{A})$ are the column, row, and kernel spaces of $\mathbf{A}$, respectively.[2]

From Eq. (E.2), $\operatorname{rank}(\mathbf{A}) = \operatorname{rank}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right)$ if and only if $\operatorname{nullity}(\mathbf{A}) = \operatorname{nullity}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right)$.

[1] Here we draw the distinction between the scatter matrix, $\mathbf{A}^\mathsf{T}\mathbf{A} \in \mathbb{R}^{m \times m}$, and the Gram matrix, $\mathbf{A}\mathbf{A}^\mathsf{T} \in \mathbb{R}^{n \times n}$. They both are symmetric and positive semidefinite but, in general, $\mathbf{A}$ does not commute with its transpose $\mathbf{A}^\mathsf{T}$, i.e., $\mathbf{A}^\mathsf{T}\mathbf{A} \neq \mathbf{A}\mathbf{A}^\mathsf{T}$.

[2] The column space (also called the range or image) of a matrix $\mathbf{A}$ is the span (set of all possible linear combinations) of its column vectors. Conversely, the row space of $\mathbf{A}$ can be defined as the column space of $\mathbf{A}^\mathsf{T}$. On the other hand, a vector $\mathbf{x} \in \mathbb{R}^m$ is in the kernel space, a.k.a. nullspace, of $\mathbf{A}$, if and only if $\mathbf{A}\mathbf{x} = \mathbf{0}_m$, where $\mathbf{0}_m$ is a column vector of zeros. Thus, the kernel space of $\mathbf{A}$ is the span of all these vectors.

To prove this, we shall show that, for any vector $\mathbf{x} \in \mathbb{R}^m$ that belongs to $\ker(\mathbf{A})$, it also belongs to $\ker\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right)$ and vice versa.

First, any $\mathbf{x} \in \ker(\mathbf{A})$ satisfies $\mathbf{A}\mathbf{x} = \mathbf{0}_m$. Therefore, $\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{0}_m$ and $\mathbf{x} \in \ker\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right)$. Second, all $\mathbf{x} \in \ker\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right)$ satisfies $\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{0}_m$. Hence, $\mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} = (\mathbf{A}\mathbf{x})^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{0}_m$. In consequence, $\mathbf{A}\mathbf{x} = \mathbf{0}_m$ and $\mathbf{x} \in \ker(\mathbf{A})$.[3] Thus, $\text{nullity}(\mathbf{A}) = \text{nullity}(\mathbf{A}^\mathsf{T}\mathbf{A})$ and $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^\mathsf{T}\mathbf{A})$, as required.

The same result applies to the Gram matrix of $\mathbf{A}$, $\mathbf{A}\mathbf{A}^\mathsf{T} \in \mathbb{R}^{n\times n}$:

$$\text{rank}(\mathbf{A}) = \text{rank}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right) = \text{rank}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right). \tag{E.3}$$

Given that $\text{rank}\left(\mathbf{A}^\mathsf{T}\right) = \text{rank}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)$ and $\text{rank}(\mathbf{A}) = \text{rank}\left(\mathbf{A}^\mathsf{T}\right)$, we obtain that $\text{rank}(\mathbf{A}) = \text{rank}\left(\mathbf{A}^\mathsf{T}\mathbf{A}\right) = \text{rank}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)$.

In consequence, $\mathbf{A}^\mathsf{T}\mathbf{A}$ is non-singular if $\text{rank}(\mathbf{A}) = m$, and $\mathbf{A}\mathbf{A}^\mathsf{T}$ is non-singular if $\text{rank}(\mathbf{A}) = n$. In ML, $\mathbf{A}$ usually corresponds to the extended design matrix, $\mathbf{\Phi}_e$. Hence, assuming m is the number of linearly independent basis functions (including the one corresponding to the bias term of the model), $\mathbf{A}^\mathsf{T}\mathbf{A}$ is particularly useful when $\text{rank}(\mathbf{A}) = m$, i.e., the number of samples is greater than or equal to the number of linearly independent basis functions. Conversely, assuming n is the number of samples linearly independent in the basis functions space, $\mathbf{A}\mathbf{A}^\mathsf{T}$ is useful when $\text{rank}(\mathbf{A}) = n$, i.e., the number of samples linearly independent in the basis functions space is lower than or equal to the number of basis functions, including the case where the basis functions are infinite-dimensional (in the Hilbert space).

E.2 Multiplication by a Non-Singular Matrix Preserves Rank

We will show that, for a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, and a non-singular matrix $\mathbf{B} \in \mathbb{R}^{m\times m}$, which are conformable:

$$\text{rank}(\mathbf{A}\mathbf{B}) = \text{rank}(\mathbf{A}). \tag{E.4}$$

Remember that $\text{rank}(\mathbf{A}) = \dim(\text{col}(\mathbf{A}))$. We shall prove that $\text{rank}(\mathbf{A}\mathbf{B}) = \text{rank}(\mathbf{A})$ if and only if $\text{col}(\mathbf{A}\mathbf{B}) = \text{col}(\mathbf{A})$.

Denote by $\mathcal{C}$ the column space of $\mathbf{A}$. Any vector $\mathbf{c} \in \mathcal{C}$ can be written as a linear

[3] By definition, the kernel space of any matrix contains the vector of zeros.

combination of the columns of $\mathbf{A}$: $\mathbf{c} = \mathbf{A}\mathbf{v}$, where $\mathbf{v} \in \mathbb{R}^m$ is the vector of coefficients of the linear combination. Since $\mathbf{B}$ is non-singular, it has m linearly independent columns that span the space of all m-dimensional vectors (they are equivalent to the canonical basis). Therefore, $\exists\ \mathbf{u} \in \mathbb{R}^m :\ \mathbf{v} = \mathbf{B}\mathbf{u}$. Thus, $\mathbf{c} = \mathbf{A}\mathbf{v} = (\mathbf{A}\mathbf{B})\mathbf{u}$.

We have proved that any vector $\mathbf{c} \in \mathcal{C}$ can be written as a linear combination of the columns of $\mathbf{AB}$. Moreover, because matrix multiplication is associative, note that for any vector of coefficients $\mathbf{u}$, if $\mathbf{c} = (\mathbf{AB})\mathbf{u}$, then $\mathbf{c} = \mathbf{A}(\mathbf{B}\mathbf{u})$. Consequently, we have proved that $\text{col}(\mathbf{AB}) = \text{col}(\mathbf{A})$, as required.

The same result applies to $\mathbf{B}\mathbf{A}^\mathsf{T}$:

$$\text{rank}\left(\mathbf{B}\mathbf{A}^\mathsf{T}\right) = \text{rank}\left(\mathbf{A}^\mathsf{T}\right) = \text{rank}(\mathbf{A}). \tag{E.5}$$

This is evident because $\text{rank}\left((\mathbf{AB})^\mathsf{T}\right) = \text{rank}\left(\mathbf{B}^\mathsf{T}\mathbf{A}^\mathsf{T}\right) = \text{rank}\left(\mathbf{B}\mathbf{A}^\mathsf{T}\right) = \text{rank}\left(\mathbf{A}^\mathsf{T}\right) = \text{rank}(\mathbf{A})$. Since $\mathbf{B}$ is a non-singular matrix, transposing it does not change the result.

E.3 Upper Bound on the Rank of the Multiplication of Two Matrices

We shall show that, for two conformable matrices $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{B} \in \mathbb{R}^{m\times p}$:

$$\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})). \tag{E.6}$$

Each column of $\mathbf{AB}$ is a linear combination of the columns of $\mathbf{A}$, and therefore $\text{col}(\mathbf{AB}) \subseteq \text{col}(\mathbf{A})$ and $\text{rank}(\mathbf{AB}) \leq \text{rank}(\mathbf{A})$. Similarly, each row of $\mathbf{AB}$ is a linear combination of the rows of $\mathbf{B}$, so $\text{row}(\mathbf{AB}) \subseteq \text{row}(\mathbf{B})$. Thus, $\text{rank}(\mathbf{AB}) \leq \text{rank}(\mathbf{B})$.

From both inequalities, we deduce that $\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$.

E.4 Upper Bound on the Rank of the Sum of Two Matrices

We shall show that the rank of the sum of conformable matrices is subadditive, i.e., for two conformable matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times m}$:

$$\text{rank}(\mathbf{A} + \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}). \tag{E.7}$$

To see this, recall that $\text{rank}(\mathbf{A}) = \dim(\text{col}(\mathbf{A}))$. It suffices to prove that the column space of $\mathbf{A} + \mathbf{B}$ is a subset of the direct sum of the column spaces of $\mathbf{A}$ and $\mathbf{B}$.[4]

In particular, let $\mathbf{y} \in \text{col}(\mathbf{A} + \mathbf{B})$ This means that there is a vector $\mathbf{x} \in \mathbb{R}^m$ for which $\mathbf{y} = (\mathbf{A} + \mathbf{B})\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{x}$. Since we can express $\mathbf{y}$ as a linear combination of vectors in the column spaces of $\mathbf{A}$ and $\mathbf{B}$, this means that $\mathbf{y} \in (\text{col}(\mathbf{A}) \oplus \text{col}(\mathbf{B}))$. Thus, $\text{col}(\mathbf{A} + \mathbf{B}) \subseteq (\text{col}(\mathbf{A}) \oplus \text{col}(\mathbf{B}))$.

From the previous result, note that $\dim(\text{col}(\mathbf{A} + \mathbf{B})) \leq \dim(\text{col}(\mathbf{A}) \oplus \text{col}(\mathbf{B})) = \dim(\text{col}(\mathbf{A})) + \dim(\text{col}(\mathbf{B})) - \dim(\text{col}(\mathbf{A}) \cap \text{col}(\mathbf{B})) = \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) - \dim(\text{col}(\mathbf{A}) \cap \text{col}(\mathbf{B}))$. We can conclude that $\text{rank}(\mathbf{A} + \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$.

[4]The direct sum of two subspaces $\mathcal{U}$ and $\mathcal{W}$, denoted as $\mathcal{U} \oplus \mathcal{W}$, is the sum of subspaces in which $\mathcal{U}$ and $\mathcal{W}$ have only the vector of zeros in common (Rosen, 1999).