Novel Methods for Multi-view Learning with Applications in Cyber Security

a thesis presented for the degree of

Doctor of Philosophy of Imperial College London

and the

Diploma of Imperial College

by

Jack Hogan

Department of Mathematics

Imperial College

180 Queen's Gate, London SW7 2AZ

April 2023

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Signed: _____

Copyright

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial-No Derivatives 4.0 International Licence (CC BY-NC-ND). Under this licence, you may copy and redistribute the material in any medium or format on the condition that; you credit the author, do not use it for commercial purposes and do not distribute modified versions of the work. 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. 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.

Novel Methods for Multi-view Learning with Applications in Cyber Security

Abstract

Modern data is complex. It exists in many different forms, shapes and kinds. Vectors, graphs, histograms, sets, intervals, etc.: they each have distinct and varied structural properties. Tailoring models to the characteristics of various feature representations has been the subject of considerable research. In this thesis, we address the challenge of learning from data that is described by *multiple* heterogeneous feature representations.

This situation arises often in cyber security contexts. Data from a computer network can be represented by a graph of user authentications, a time series of network traffic, a tree of process events, etc. Each representation provides a complementary *view* of the holistic state of the network, and so data of this type is referred to as *multi-view* data. Our motivating problem in cyber security is *anomaly detection*: identifying unusual observations in a joint feature space, which may not appear anomalous marginally.

Our contributions include the development of novel supervised and unsupervised methods, which are applicable not only to cyber security but to multi-view data in general. We extend the generalised linear model to operate in a vector-valued reproducing kernel Hilbert space implied by an operator-valued kernel function, which can be tailored to the structural characteristics of multiple views of data. This is a highly flexible algorithm, able to predict a wide variety of response types. A distinguishing feature is the ability to simultaneously identify outlier observations with respect to the fitted model. Our proposed unsupervised learning model extends multidimensional scaling to directly map multi-view data into a shared latent space. This vector embedding captures both commonalities and disparities that exist between multiple views of the data. Throughout the thesis, we demonstrate our models using real-world cyber security datasets.

Acknowledgments

First, I would like to thank my supervisor, Professor Niall Adams. Without his encouragement, I would not have undertaken this Ph.D. and without his incredible support and patience, I doubtless would never have finished. Niall has been a constant source of ideas, advice and good humour, and I'm extremely grateful for all his guidance.

I am also thankful for the scholarship provided by QinetiQ. This Ph.D. would not have been possible without their financial support.

In 2019, I was fortunate to work on a research project for Rolls Royce and the Alan Turing Institute. I'm grateful to Dr Andrew Duncan for the opportunity and for all his guidance during the project.

Throughout my time at Imperial, I've had great office-mates, who made my experience immeasurably more enjoyable. I've also received helpful advice, ideas and comments from many people; in particular, I'd like to thank Professor David Hand and Dr Christoforos Anagnostopoulos for helpful comments on a manuscript that forms the basis for Chapter 4 of this thesis. Andy Thomas was always quick to provide technical assistance with software packages and computational resources. Thanks also to Ed Cohen, Dean Bodenham, Nick Heard, Mark Briers and Josh Neil for enjoyable discussions over the years.

Finally, a special thank you to all my family and friends; in particular, to my father for all his advice and encouragement, to my mother for always thinking of me, and to Jasmine, for her endless patience and support.

Jack Hogan

Contents

Notation . . . 1

1 Introduction **3**
1.1 Contributions and Thesis Outline . . . 5
1.2 Publications . . . 7

2 Background **8**
2.1 Multi-view Data . . . 8
2.2 Multi-view Learning . . . 13
2.2.1 Multi-view Outlier Detection . . . 15
2.3 Cyber Security Datasets . . . 16
2.3.1 LANL Dataset . . . 16
2.3.2 EMBER Malware Dataset . . . 19

3 Computer Network Data Analysis **22**
3.1 Statistical Cyber Security . . . 23
3.2 Supervised Learning using Time-shifted Labels . . . 25
3.2.1 Time-shifted Label . . . 26
3.2.2 Data Fusion . . . 26
3.2.3 Entity Fusion . . . 28
3.3 Experiment . . . 28
3.4 Discussion . . . 31

4 On Averaging ROC curves **32**
4.1 Background and Relevant Literature . . . 33
4.2 ROC Graphs . . . 35
4.2.1 Definition . . . 36
4.2.2 Estimation . . . 37
4.2.3 Area Under the ROC Curve . . . 38
4.3 Averaging ROC Curves . . . 38
4.3.1 Pooling . . . 41
4.3.2 Vertical Averaging . . . 41
4.3.3 Threshold Averaging . . . 42
4.4 Interpreting Average ROC Curves . . . 42
4.4.1 ROC Space Averaging . . . 44
4.5 Simulated Example . . . 45

4.6 Discussion 48

5 Semi-supervised Learning via Co-regularised Kernel Logistic Regression **49**
5.1 Background and Relevant Literature 50
5.2 Reproducing Kernel Hilbert Spaces 52
5.2.1 Scalar-valued Kernel Machines 52
5.2.2 Vector-valued RKHSs 54
5.2.3 Multi-view Kernels 55
5.3 Multi-view Semi-supervised Structural Risk Minimisation . . . 56
5.3.1 Supervised Learning Framework 56
5.3.2 Relation to Single-view Learning Schemes 58
5.3.3 Semi-supervised Learning 58
5.4 Semi-supervised Kernel Logistic Regression 61
5.5 Experiments 63
5.5.1 Simulated Data 63
5.5.2 Malware Detection 66
5.6 Discussion 70

6 Multi-view Generalised Kernel Machines **72**
6.1 Background and Relevant Literature 74
6.2 A Unified Learning Scheme 76
6.2.1 Generalised Linear Models 76
6.2.2 Multi-view GKMs 78
6.2.3 Parameter Estimation 79
6.2.4 Algorithm 81
6.2.5 Prediction 82
6.2.6 Outlier Detection 82
6.3 Multi-view Kernel Functions 86
6.3.1 MKL Kernel 87
6.3.2 Cross-covariance Kernel 87
6.3.3 Learned Metric Kernel 88
6.4 Example Response Types 89
6.4.1 Logistic Regression 89
6.4.2 Poisson Regression 90
6.4.3 Multinomial Logistic Regression 91
6.5 Experiments 93
6.5.1 Binary, Multi-class and Poisson Regression 94
6.5.2 Anomaly Detection 99
6.6 Discussion 102

7 Multi-view Multidimensional Scaling **103**
7.1 Background and Relevant Literature 105
7.2 Multidimensional Scaling 107
7.2.1 The SMACOF Algorithm for MDS 108
7.3 Multi-view Embedding 109
7.3.1 Multi-view MDS 110
7.3.2 Extending SMACOF for Multi-view MDS 112
7.4 Experiments 114
7.4.1 Simulated Data 114
7.4.2 Cyber Situational Awareness 118
7.5 Discussion 127

8 Conclusion **128**

References **153**

Notation

Unless stated otherwise, plain x represents a scalar, boldface $\boldsymbol{x}$ represents a vector (or symbolic data—see Section 2.1), calligraphic $\mathcal{X}$ represents a topological space and capitalised boldface $\boldsymbol{X}$ represents a matrix.

List of Symbols

tp	true positive rate
fp	false positive rate
P, N	positive and negative subsets of binary dataset
k	scalar-valued kernel function
K	operator-valued kernel function
$\boldsymbol{K}$	kernel gram matrix
$\langle \cdot, \cdot \rangle_{\mathcal{H}}$	inner-product in $\mathcal{H}$
$\mathcal{H}_k$	RKHS of the reproducing kernel k
$\mathcal{L}(\mathcal{Z})$	space of linear functions from $\mathcal{Z}$ to itself
$J(\cdot)$	SRM objective function
$V(\cdot, \cdot)$	loss function
$\|\cdot\|^2$	squared L^2-norm
$\otimes$	Kronecker product
$\boldsymbol{I}_n$	$n \times n$ identity matrix
$\mathbf{1}_n$	n-dimensional vector of ones
$\boldsymbol{\nabla}_{\boldsymbol{\alpha}}$	gradient with respect to $\boldsymbol{\alpha}$
$\boldsymbol{\nabla}^2_{\boldsymbol{\alpha}}$	second derivative, i.e. Hessian matrix with respect to $\boldsymbol{\alpha}$
$\ell(\cdot)$	log-likelihood function
δ_{ij}	dissimilarity between observations i and j
$\boldsymbol{\Delta}$	matrix of pairwise dissimilarities
$d_{ij}(\boldsymbol{Z})$	Euclidean distance between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$
$\mathrm{tr}(\cdot)$	matrix trace
$\mathbb{O}^{p \times q}$	space of semi-orthogonal $p \times q$ matrices

List of Acronyms

LANL	Los Alamos National Laboratory dataset (Section 2.3.1)
WLS	Windows Logging Service—subset of the LANL dataset
EMBER	Malware dataset (Section 2.3.2)
ROC	Receiver Operating Characteristic
AUC	Area Under the (ROC) Curve
RKHS	Reproducing kernel Hilbert space
SRM	Structural Risk Minimisation
SVM	Support Vector Machine
tf-idf	term frequency times inverse-document frequency
GLM	Generalised Linear Model
GKM	Generalised Kernel Machine
MVGKM	Multi-view Generalised Kernel Machine
MKL	Multiple Kernel Learning
MVML	Multi-view Metric Learning
MAE	Mean absolute error
MDS	Multidimensional scaling
PCA	Principal Component Analysis
SMACOF	Scaling by Majorising a Complex Function

1

Introduction

The most common approach to statistical modelling and machine learning problems is to assume that data are represented in a single vector space. Modern data, however, is multiform: it exists in many forms, shapes and kinds; for example, lists, strings, histograms, graphs, trees, etc. Often, each instance in a dataset will have distinct representations from multiple—and potentially disparate—feature spaces. Each representation can be considered as providing a complementary *view* about the underlying object and hence data of this type are often referred to as *multi-view* data. Archetypal examples include images, which can be described by colour and texture features; biological samples, represented by genome sequence, DNA, metabolic type etc.; and video segments, represented by the image content and the audio content.

As advances in technology allow data to be gathered inexpensively from multifarious data sources, the natural and ubiquitous occurrence of multi-view data has prompted interest in so-called *multi-view learning*. Commonly, in order to accommodate multi-view data in a statistical or machine learning setting, one of two crude approaches is used: *early fusion* calls for the data from separate views to be simply combined or concatenated into a single fea-

ture representation prior to modelling; *late fusion* involves individual models being developed per view, with their outputs then combined or averaged. In Chapter 2, we will discuss a number of reasons why both approaches may have unsatisfactory consequences on the quality of the resulting model. Broadly, the former can lead to valuable information being forfeited when summarising data into a common representation, while the latter is unable to capture correlations among features from different views. Our goal in this thesis is to develop more sophisticated approaches that can maximise the information captured from complex multi-view data.

Cyber security is one application area for which multi-view learning has particular promise. This is an extremely data-rich domain: vast datasets referring to numerous types of network or computer events are routinely available within even small enterprises, e.g. network data such as Netflow traffic, web cache and proxy logs, domain name service (DNS) lookup records (Kent, 2016; Morgan et al., 2016), as well as host-based data such as authentication logs, process trees and command histories (Glass-Vanderlan et al., 2018; Sanna Passino et al., 2023b). Traditionally, cyber security systems such as firewalls and anti-virus software have relied on *rule-based* mechanisms. These can be thought of as decision trees with hard-coded parameters, often extracted through trial and error and expert knowledge, and taken from real-world examples (Hero et al., 2023). While such methods are successful at detecting repeat attacks, a major disadvantage is that they are unable to detect malicious activity or attack behaviour for which a rule, or *signature*, has not been previously created. There is a need for additional systems, based on statistical modelling of network data, to complement existing cyber defence systems and enhance the abilities of cyber security professionals.

In contrast to rule-based detection, statistical approaches attempt to characterise the normal behaviour of some aspect of a computer network and subsequently to detect anomalous departures from normal behaviour (Bhuyan et al., 2014; Mukherjee et al., 1994; Ye et al., 2002). In this way, *anomaly detection* does not explicitly identify malicious behaviour, but rather locates unusual and surprising activity, which may be symptomatic of an intrusion. When designing signature-based detection methods, human experts typically

try to "combine" different attack characteristics in order to obtain low false positive rates and high attack detection rates (Giacinto et al., 2003). A statistical anomaly detector should do the same. By capturing mutual relations between multiple data sources, multi-view approaches may be able to identify the unusual *co-occurence* of observations across data sources, which may not be unusual within any individual data source or representation. The very last part of this thesis demonstrates precisely this capability.

The two main challenges addressed in this thesis are thus: (1) how to combine information contained in different complicated data structures in a synergistic way for various learning problems; and (2) how to detect anomalous or outlying data points with respect to co-occurring signals across different views of data, which may not appear anomalous in isolation.

Although cyber security is the key motivator for the procedures we develop to address these two challenges, they are not limited to this domain. In Chapters 5 to 7, we consider separately the tasks of semi-supervised, supervised and unsupervised learning, and the methods we develop are applicable in each case to multi-view data in general.

1.1 Contributions and Thesis Outline

This thesis provides novel contributions to both the fields of multi-view learning and cyber security. From a multi-view learning perspective, we make two major contributions: we propose novel frameworks for supervised (Chapter 6) and unsupervised (Chapter 7) learning using complex, multi-view data. In both cases, we derive optimisation algorithms and demonstrate performance on real-world data. From a methodological perspective of statistical cyber security, we propose several novel approaches, including a strategy for novelty detection (Chapter 3), a tool for investigating port scanning (Chapter 6) and a framework for asset monitoring and situational awareness (Chapter 7). We next outline the structure and contributions of the remainder of the thesis.

In Chapter 2, we discuss background material, beginning with a detailed description of what constitutes *multi-view data*. The prevalence of such data in

modern applications is highlighted through a series of examples. We describe the shortcomings of simplistic or naïve approaches to modelling multi-view data and highlight desirable properties for a multi-view learning method. Two cyber security datasets that are used in the thesis are introduced.

In Chapter 3, we address a prevalent problem in cyber security that precludes the use of supervised learning approaches: there is a dearth of ground-truth labels in network data to indicate benign and malicious activity. We propose a framework for contriving labels from the data so that classification techniques can be deployed for novelty detection. In presenting this framework, we explore the use of early and late fusion and highlight the sub-optimality of both approaches.

When comparing early and late fusion classifiers, we measure the difference in their *average* performance using ROC curves. There are many ways to reason about how an average ROC curve should be constructed, though little guidance is provided in the literature. To address this, in Chapter 4 we take a brief detour from multi-view learning and explore the interpretation of different methods of averaging ROC curves. We perform simulation studies to illustrate the dangers of choosing the incorrect method for a given study.

Fundamentally, the quality of any statistical model hinges on its ability to capture the underlying structure of a dataset. Kernel methods offer a powerful paradigm for learning non-linear patterns in diverse data representations. In Chapter 5, we exploit a semi-supervised learning framework based on operator-valued kernel functions to simultaneously address challenges presented by multi-view data and also the scarcity of labels in cyber security problems such as malware detection. We modify the loss function of an existing framework to develop a probabilistic classifier that can learn from partially labelled multi-view data.

In Chapter 6, we use this framework as the starting point to propose a novel general-purpose multi-view supervised learning scheme. We extend the classical generalised linear model to operate in the vector-valued reproducing kernel Hilbert space implied by an operator-valued multi-view kernel. Our framework has a simple optimisation procedure for binary, categorical,

integer- and continuous-valued responses, it results in interpretable outputs and comes with diagnostic capabilities such as outlier detection. We demonstrate its performance on a variety of datasets, including an outlier detection example in the cyber security setting.

The model developed in Chapter 6 learns patterns within multi-view data based on their representation in a high-dimensional space implied by the multi-view kernel. In Chapter 7, we propose a novel way of explicitly learning a vector representation of multi-view data in a shared latent space. Unsupervised learning and anomaly detection can then be performed directly using the embeddings of the data in this space. We describe how such an approach could be used for cyber security situational awareness and provide a demonstration using real data.

The thesis concludes in Chapter 8 with a summary of contributions.

1.2 Publications

This thesis contains research that has either been accepted for publication or is under review for publication at the time of writing.

Chapter 3 is based on the following paper:
J. Hogan and N. M. Adams. A study of data fusion for predicting novel activity in enterprise cyber-security. In *IEEE International Conference on Intelligence and Security Informatics*, pages 37–42, 2018.

Chapter 4 is based on the following paper:
J. Hogan and N. M. Adams. On averaging ROC curves. *Transactions on Machine Learning Research*, 2023.

Chapter 6 is based on the following paper, currently under revision:
J. Hogan and N. M. Adams. Multi-view generalised kernel machines for regression, classification and outlier detection. *Journal of Machine Learning Research*, 2023.

2

Background

This chapter provides a summary of background material for the thesis. We begin with a discussion of multi-view data in Section 2.1, highlighting its salient characteristics and illustrating its ubiquity in modern data applications. In Section 2.2, we briefly discuss some of the challenges that are presented when constructing statistical models or machine learning algorithms for multi-view data. We outline the shortcomings of simplistic approaches and motivate the methods that we develop in this thesis. Section 2.3 details two cyber security datasets that will be used: one recording network traffic and authentication behaviour within an enterprise network and the other describing a collection of benign and malicious computer programmes.

2.1 Multi-view Data

In this thesis, we are concerned with a specific characteristic of certain data; that is, *the ability for the total set of data per subject (i.e. features) to be naturally decomposed into distinct subsets, each with heterogeneous statistical properties.* It could be argued that this criterion is met by, and therefore encompasses, essentially all data comprising more than one feature. How-

ever, in this thesis, we place particular emphasis on the latter condition: that the subsets of features are in some way heterogeneous or *structurally diverse*. In practice, this decomposition of features is most often encountered when a dataset is *composed* of sets of measurements recorded from multiple different sources, or sets of features engineered for different purposes. For example, in a dataset of captioned images, the total data available to describe each subject comprises image data and text data. As we will see, there are many different families of dataset, which may appear very different—be it conceptually or constructionally—yet they share this relevant characteristic.

Across different application domains and learning settings, data of this type can be found referred to by various names:

- *Multi-source* (Ouyang et al., 2014; Zhang et al., 2018) is often used when a given dataset was generated from different sources or measurement devices; e.g. genomes and proteomes measured from biological specimens.
- *Multi-modal* (Ngiam et al., 2011; Srivastava and Salakhutdinov, 2012) is common in the computer vision literature, where different feature sets may be available that represent different *modalities*; e.g. image and audio.
- *Multi-feature* (Scheirer and Slaney, 1997; Yang et al., 2012) typically refers to datasets comprising multiple extracted feature sets from a single source of raw data; e.g. shape and texture features extracted from the pixel intensities of an image.
- *Multi-view* (Bickel and Scheffer, 2004; Kumar et al., 2011) is a general term to describe data that has multiple feature sets that each capture a different *view* or aspect of the subject.

In our opinion, the term *multi-source* places unnecessary emphasis on the property that different feature sets are derived from different instruments; while such data may have heterogeneous structural representations or statistical properties, this is not necessarily the case. *Multi-modal* does convey

the intuition that different feature sets represent different sensory inputs or channels of information; however, multiple feature representations may be available corresponding to a single modality. Additionally, confusion may arise as the term is already commonly used to describe probability distributions with multiple peaks. Similarly, the term *multi-feature* data may be confused with multivariate data. We favour the term *multi-view*, as we believe it captures the notion that different feature sets—irrespective of how they were generated—may offer different but complementary information regarding a subject.

In the remainder of this thesis, let the feature space of multi-view data be given by $\mathcal{X} = \mathcal{X}^{(1)} \times \ldots \times \mathcal{X}^{(v)}$, with $\mathcal{X}^{(s)}, \mathcal{X}^{(t)}$ possibly disjoint spaces for $s \neq t$. We assume that for each view $s \in \{1, \ldots, v\}$, the observations $\boldsymbol{x}_1^{(s)}, \ldots, \boldsymbol{x}_n^{(s)}$ are independent realisations of a random variable $X^{(s)}$. Then the ith sample is given by $\boldsymbol{x}_i = \{\boldsymbol{x}_i^{(s)}\}_{s=1}^{v}$ and the sth view of the dataset is given by $\boldsymbol{x}^{(s)} = \{\boldsymbol{x}_i^{(s)}\}_{i=1}^{n}$. Note that for simplicity, we use vector notation to denote the sample $\boldsymbol{x}_i^{(s)}$, though it may be so-called *symbolic data*; i.e. any arbitrary structure, such as a graph, histogram, tree etc. Additionally, where samples from a specific view are represented as vectors, it is not required that they share the same dimensionality; for example, variable-length strings or time series. For a detailed description of symbolic data, along with numerous examples, see e.g. Billard and Diday (2003); Bock and Diday (1999).

The motivating application of this thesis is cyber security, and in Section 2.3, we describe two cyber security datasets and highlight multiple heterogeneous views that can be observed in each, including numeric vectors, graphs, histograms, trees and lists. To illustrate the prevalence of multi-view data beyond the motivating example of cyber security, we provide the following additional examples:

Biological Datasets

Data encountered in biological studies offer a perfect example of multi-view data: gene expression data are represented as vectors or time series; protein–protein interactions can be expressed as graphs; gene sequence data are typically strings from a 4-symbol alphabet, while protein sequences are

strings from a 20-symbol alphabet (Lanckriet et al., 2004b).

For example, Serra et al. (2015) used data from the Memorial Sloan Kettering Cancer Center genomics data portal[1] to cluster cancer patients in order to identify novel disease subgroups. Each sample in the dataset corresponds to a different prostate cancer tumour and four views are used: clinical data features, gene expressions, microRNA expressions and copy number variations.

Image Datasets

In image recognition and classification tasks, multi-view data abound. For example, each sample in a dataset may be represented by multiple images, or each image may be represented by multiple feature vectors, or both.

Multi-feature datasets are particularly common, or may easily be constructed from raw image datasets. The *Multiple Features* dataset (van Breukelen et al., 1998), available from the UCI machine learning repository[2], comprises multiple feature representations engineered from a collection of binary images of handwritten numerals $(0-9)$, which were extracted from Dutch utility maps. Six different views (i.e. feature vectors) are provided for each sample: (1) 76 Fourier coefficients of the character shapes; (2) 216 profile correlations; (3) 64 Karhunen-Love coefficients; (4) 240 pixel averages in 2×3 windows; (5) 47 Zernike moments; (6) 6 morphological features.

Similarly, the *Animals with Attributes* dataset[3] (Lampert et al., 2009) comprises coloured photographs of 50 different species of animals. Six views are available: (1) 2,688 colour histogram features; (2) 2,000 self-similarity features; (3) 252 pyramid histogram of oriented gradients features (PHOG); (4) 2,000 scale-invariant feature transform values (SIFT); (5) 2,000 colour SIFT values; (6) 2,000 speeded-up robust features (SURF).

Multi-view datasets are also encountered when images are captured using various devices or using various light wavelengths. For example, *hyperspectral*

[1] https://cbio.mskcc.org/cancergenomics/prostate/data/
[2] https://archive.ics.uci.edu/ml/datasets/Multiple+Features
[3] https://cvml.ista.ac.at/AwA/

imaging can be used by geologists to obtain aerial photographs of an area of land using a camera that can capture light from multiple bands of the electromagnetic spectrum beyond those visible to the human eye. *Indian Pines*[4] (Baumgardner et al., 2015) is a hyperspectral image segmentation dataset capturing a single landscape in Indiana, U.S.A. across 200 spectral bands. Ground truth labels indicate segments of the scene that are covered by named crops, woodland or highways.

A further source of multi-view image datasets is the capturing of multiple images at different angles or light conditions. For example, the Columbia object image library (*COIL-100*) dataset[5] (Nene et al., 1996) is a collection of images of 100 objects, each photographed at 72 fixed angles.

Text Datasets

When analysing text, there are a number of ways in which the data may be considered multi-view.

The *Reuters Multilingual Corpus* dataset[6] (Amini et al., 2009) comprises over 100,000 articles from six classes; each article is provided in five languages: English, French, German, Italian, and Spanish.

Text datasets may also be multi-modal in nature, represented in both a feature space describing the raw text and a graph space describing relationships between texts. For example, a number of well-known datasets have been constructed from scientific articles; see e.g. *Cora*[7] (Sen et al., 2008), *PubMed*[8] (Namata et al., 2012), *WebKB*[9] (Lu and Getoor, 2003). In each, the text of the article is represented as a bag-of-words feature vector and a graph contains links between articles if one article cites the other.

[4] https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes

[5] https://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php

[6] https://archive.ics.uci.edu/ml/datasets/Reuters+RCV1+RCV2+Multilingual,+Multiview+Text+Categorization+Test+collection

[7] https://relational.fit.cvut.cz/dataset/CORA

[8] https://linqs.org/datasets/#pubmed-diabetes

[9] https://linqs.org/datasets/#webkb

Multimodal Datasets

Finally, it is worth noting that, beyond the three described here, there are many more different types of data for which multi-view features may be available. Further, it is often the case that multiple different types of data are available per subject in a dataset, *each with potentially multiple views*. For example, a number of datasets are available containing images and associated text captions for use in object detection; see e.g. *WIT*[10] (Srinivasan et al., 2021), *COCO*[11] (Lin et al., 2014). For emotion detection, the *IEMOCAP* dataset[12] (Busso et al., 2008) contains visual, audio and text views, along with annotations indicating the presence of 9 emotions (angry, excited, fear, sad, surprised, frustrated, happy, disappointed and neutral).

Later in this thesis, we will consider datasets that combine strings and histograms, multiple audio features, images and text, among others.

2.2 Multi-view Learning

Given the varied nomenclature used to describe multi-view data, it is unsurprising that there also exists many terms for describing statistical and machine learning methods or approaches for dealing with multi-view data; e.g. *data fusion* (Lanckriet et al., 2004b; Žitnik and Zupan, 2015), *data integration* (Gligorijević and Pržulj, 2015; Subramanian et al., 2020), *multi-modal learning* (Huang et al., 2021; Ngiam et al., 2011; Srivastava and Salakhutdinov, 2012), *multi-view learning* (Sindhwani and Rosenberg, 2008; Zhao et al., 2017b). In each, the challenge to be addressed is how to combine information contained in multiple, heterogeneous feature representations of each subject in a dataset. In the language of data fusion, various approaches are often categorised as *early fusion* or *late fusion*, referring to the stage in the modelling process during which information from different views is combined.

When the various views or modalities of a dataset are highly heterogeneous, a common but naïve approach to statistical modelling or machine learning

[10] https://github.com/google-research-datasets/wit
[11] https://cocodataset.org
[12] https://sail.usc.edu/iemocap/index.html

is to develop individual models per view and then combine or average their outputs (e.g. Long et al. (2008); Serra et al. (2015)). This simple late fusion strategy has the advantage that the individual models can be tailored to the statistical properties of each view's structural representation. For example, in a classification task using captioned images, a convolutional neural network could be trained using the image data and a support vector machine could be applied to the text data. However, as each model is trained in isolation, correlations among features from different views are ignored.

An alternative approach is to simply combine or concatenate the data from separate views into a single feature representation prior to model fitting. In principle, such an early fusion model is able to capture relationships among features from within and between views. However, when the distributions and scales of feature spaces are very different, it may be very difficult to discover the highly non-linear relationships that exist between low-level features across views (Ngiam et al., 2011). Consider captioned images again as an example: the text view could be represented as discrete sparse word-count vectors, whereas the image view is typically represented using pixel intensities, which are real-valued and dense. This makes it much more challenging to discover relationships across views and tends to lead to over-fitting (Srivastava and Salakhutdinov, 2012). Another problem is the information loss incurred when combining data from heterogeneous representations into a common representation (e.g. combining a graph structure with a histogram). Often, we resort to summarising the data from different views into numeric vectors for concatenation, forfeiting much of the information contained in their natural structural representation.

A trade-off is apparent: late fusion prioritises capturing the potential signal contained within each view's natural structural representation, while early fusion prioritises unlocking potential signal by combining information between views. This compromise is summarised in Table 2.1. The goal of multi-view learning, and our motivation in this thesis, is to develop methods that combine the benefits of both strategies.

Early efforts in this vein were inspired by the way in which humans perceive objects using a combination of senses (de Sa, 1994a,b). For example,

	Capture maximum information within each view	Capture correlation between views
Early fusion	✗	✓
Late fusion	✓	✗

Table 2.1: The trade-off between early and late data fusion strategies for multi-view learning.

de Sa and Ballard (1998) developed an unsupervised algorithm capable of learning from co-occurring patterns of sound signals and lip motion from a human speaker by training separate neural networks on the signals from the two views and minimising the fraction of training samples on which the two networks disagree. Similar ideas can be found in Becker (1996); Becker and Hinton (1992), which led to the development of *co-training* (Blum and Mitchell, 1998) and *co-regularisation* (Sindhwani et al., 2005) approaches to multi-view learning. In each, the late fusion strategy of training separate models per view is endowed with the ability to capture dependence between views. We build on the same idea for the models we develop in Chapters 5 and 6. In Chapter 7, we go in the opposite direction, adopting an early fusion strategy of mapping all views to a shared representation, and endowing it with the ability to maximise the information captured from each view.

2.2.1 Multi-view Outlier Detection

The identification of unusual, surprising or anomalous observations within a dataset is an important challenge in many fields of data analysis; for example, fraud detection, system quality control, process monitoring, intrusion detection, among others. Following seminal studies by Chauvenet (1863); Dixon (1953); Grubbs (1969), the first comprehensive treatment of outliers in statistical data was provided by Barnett and Lewis (1978), who define an outlier as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data". A wide range of techniques have been developed for single-view outlier detection, which may be density-based (Breunig et al., 2000; Knorr et al., 2000), correlation-based (Kriegel et al., 2012), clustering-based (He et al., 2003), among others (for a

recent review, see Boukerche et al. (2020)).

In this thesis, we develop techniques that can be used for *multi-view outlier detection*. This is an emerging set of methods (see e.g. Das et al. (2010); Marcos Alvarez et al. (2013); Zhao et al. (2017a)) that attempt to simultaneously leverage multiple complementary views of a dataset to more accurately and robustly identify unusual observations. For example, individual views of a given observation may not appear anomalous in isolation, yet their co-occurrence may be highly unusual; conversely, an observation that may appear anomalous in one view may be explained by co-occurring attributes in other views.

2.3 Cyber Security Datasets

In this section, we describe two cyber security datasets that will be used in this thesis, illustrating the heterogeneous nature of the data structures contained within each. Beyond the obvious need to handle complex data, these datasets highlight additional modelling challenges that are presented by cyber security problems: when modelling the LANL enterprise network data described below, there is a requirement to identify outliers as part of the modelling process, while malware detection calls for an ability to learn from scarce amounts of labelled data. Other sources of cyber security data exist (e.g. session data (Highnam et al., 2021; Sanna Passino et al., 2023b)) but are not considered in this thesis.

2.3.1 LANL Dataset

Throughout this thesis, we use the publicly available 'Unified Host and Network Data Set' (Turcotte et al., 2018), released by Los Alamos National Laboratory (LANL)[13]. This comprises two complementary sets of data collected from the LANL enterprise network over the course of approximately 90 days, describing inter- and intra-device activity respectively. Privacy and security concerns demand that IP addresses, hostnames and usernames

[13]The data is available at `https://csr.lanl.gov/data/2017`

Name	Description
Time	The start time of the event
Duration	The duration of the event in seconds
SrcDevice	The device that likely initiated the event
DstDevice	The receiving device
Protocol	The protocol number
SrcPort	The port used by the *SrcDevice*
DstPort	The port used by the *DstDevice*
SrcPackets	The number of packets the *SrcDevice* sent during the event
DstPackets	The number of packets the *DstDevice* sent during the event
SrcBytes	The number of bytes the *SrcDevice* sent during the event
DstBytes	The number of bytes the *DstDevice* sent during the event

Table 2.2: Netflow data measurements, reproduced from Turcotte et al. (2018).

be anonymised before any real-world enterprise data can be made publicly available. A distinguishing feature of this collection of data is that the (anonymised) identity of devices is consistent within and between datasets, wherever possible. Event times also align across the two datasets, facilitating statistical models that take advantage of both datasets simultaneously.

Netflow Dataset

The network flow (commonly referred to as *Netflow*) dataset contains information collected at router level. Each *flow* between two IP addresses is an aggregate summary of packets (i.e. data) transferred using the same ports, under the same protocol. Approximately 200 million such flows are recorded per day in the dataset. Each record is a tuple of 11 measurements, as described in Table 2.2. In the statistical cyber security literature, a number of flow-based techniques have been proposed and shown to be successful at detecting malicious network behaviours. Methods to detect network traversal have been designed using scan statistics (Neil et al., 2013), graph theory (Djidjev et al., 2011) correlation of spatial and temporal network state information (Talpade et al., 1999) and multilevel modelling (Lawson et al., 2014).

```
{"UserName": "Comp333637$", "EventID": 4624, "LogHost": "Comp081330",
 "LogonID": "0x66deb85", "DomainName": "Domain001",
 "LogonTypeDescription": "Network", "Source": "Comp333637",
 "AuthenticationPackage": "Kerberos", "Time": 259200, "LogonType": 3}

{"UserName": "Comp654002$", "EventID": 4688, "LogHost": "Comp654002",
 "LogonID": "0x3e7", "DomainName": "Domain001",
 "ParentProcessName": "Proc247259", "ParentProcessID": "0x920",
 "ProcessName": "cscript.exe", "Time": 259200, "ProcessID": "0x18a0"}
```


Example 2.1: Two example WLS data records.

The Netflow data can be represented as a dynamic graph, with IP addresses (i.e. devices) corresponding to nodes and edges representing communications between devices. Multiple views of the data are available; for example, individual nodes in the graph can be described by their list of neighbours, while individual edges can be represented as a time series of data transfers, etc.

WLS Dataset

The WLS (Windows Logging Service) dataset contains information relating to activity taking place on each machine from the subset of devices running the Microsoft Windows operating system. Each log record describes the occurrence on a computer of one of 20 unique events. These can be subdivided into authentication, process and system events. Example events include successful/failed log-ons, processes starting/ending, Windows starting up/shutting down (see Turcotte et al. (2018) for the full list). Associated with each event is a set of attributes providing additional information; which attributes are included in a record depends on the event that was logged. This is an extremely rich dataset, providing many opportunities for cyber security researchers. As Anthony (2013) points out, it is at the host level that most initial compromises happen, and indeed where many of the proceeding steps of the attack life cycle take place.

Two example records from the WLS dataset are shown in Example 2.1. In

the first, a successful network authentication event is being recorded on the device `Comp081330`, originating from `Comp333637`; i.e. a user is logging into a computer from another computer on the network. In the second example, the process `cscript.exe` is started on `Comp654002` by user `Comp654002$`. This process was triggered by a parent process, `Proc247259`.

Again, multiple views of this data can be considered. For example, the process data could be represented as a tree structure, while the authentication events could be represented as a directed graph; each computer can be described by the list or set of processes that it runs, or by the set of users that authenticate from or to it.

2.3.2 EMBER Malware Dataset

For malware classification, we will make use of the Elastic Malware Benchmark for Empowering Researchers (EMBER) dataset[14] (Anderson and Roth, 2018). This labelled dataset contains features extracted from 1.1M binary files. For each file, eight groups of raw features are available, including both file-specific parsed features such as lists of imported and exported functions, as well as format-agnostic feature vectors and histograms. In this thesis, we will consider three different feature modalities, which we describe below:

Numeric Features

Two groups of raw features include simple numeric and categorical count values. This includes general file information, such as file size, number of imported and exported functions, presence/absence of a debug section, number of symbols, among others. Additionally, a vector of summary statistics about the file's printable strings is provided; for example, number of strings, average string length, number of urls, etc. The combined set of all numeric features is described in Table 2.3.

[14]The data is available at `https://github.com/elastic/ember`

Name	Description
size	The size of the file in bytes
vsize	The virtual size of the file
has_debug	Whether the file has a debug section
exports	Number of exported functions
imports	Number of imported functions
has_relocations	Whether the file has relocations
has_resources	Whether the file has embedded resources
has_signature	Whether the file has a signature
has_tls	Whether the file has thread local storage
symbols	Number of symbols
numstrings	Number of strings
avlength	Average string length
printables	Number of printable characters across all strings
entropy	Entropy of printable characters across all strings
paths	Number of strings that begin with `C:\\`
urls	Number of strings that begin with `http://` or `https://`
registry	Number of occurrences of `HKEY_`
MZ	Number of occurrences of `MZ`

Table 2.3: Numeric features extracted from the EMBER dataset.

Histograms

For each file, two histograms are provided: the byte histogram contains 256 integer values representing the counts of each byte value within the file, while the byte entropy histogram approximates the joint distribution $p(H, X)$ of entropy H and byte value X—see Anderson et al. (2012) for details. For our experiments in this thesis, we normalise the byte histogram to a distribution; the total file size in bytes is already represented as a feature in the numeric modality.

Strings

The raw strings representing all the imported and exported functions are also reported per file. Imported functions are grouped by library in *JSON* format and exported functions are provided as a list. We combine both sets of functions into a single list of strings per file by representing each imported function as a `library:FunctionName` pair and concatenating; an example is

```
['kernel32.dll:GetTickCount', 'shell32.dll:SHGetMalloc',
'user32.dll:ShowWindow', 'CreateAdaptor', ...]
```

Example 2.2: An example list of imported and exported functions from the EMBER dataset.

shown in Example 2.2.

3

Computer Network Data Analysis

In this chapter, we present cyber security as an emerging application area for data science in general and multi-view learning in particular. We discuss a number of challenges to the successful deployment of machine learning methods and propose a framework designed to overcome the most major of these—the lack of labelled training data. This framework is illustrated through experiments using real-world enterprise network data. The content of this chapter is adapted from Hogan and Adams (2018) and serves to motivate the more sophisticated multi-view learning approaches developed in Chapters 5 to 7.

The remainder of the chapter proceeds as follows: In Section 3.1, we discuss the growing need for effective cyber security solutions, highlighting both the promise of statistical and machine learning approaches and also the obstacles hindering their successful utilisation. In Section 3.2, we propose a framework for contriving labelled data from network traffic logs, which allow supervised learning techniques to be used for anomaly detection. In this setting, we consider the importance of data fusion—both between views relating to each

individual computer and between computers themselves. Section 3.3 presents an experiment using the LANL enterprise network dataset. This experiment, using receiver operating characteristic (ROC) curves to assess performance, reveals an issue with the ROC methodology, which motivates the next chapter.

3.1 Statistical Cyber Security

As the world becomes more interconnected, cyber attacks such as network intrusions, data theft and malicious damage to crucial systems are increasing in both prevalence and severity. An important task in cyber security is the detection of malicious activity on computer networks, either in the form of an unlawful intrusion to the network or errant behaviour within the network. For many years, enterprise cyber security systems have relied heavily on *signatures*: precise descriptions of known malicious behaviour. The dominant approach has been to design rules-based systems to monitor and scan for these indicators, which are manually defined based on knowledge gained from previously encountered and analysed attacks. Examples include file hashes, flagged IP addresses and domains, and traffic characteristics of a known Command and Control attack protocol (Morgan et al., 2016). While such methods are successful at detecting repeat attacks, they can easily be evaded; for example, making a minor modification to a malicious program will alter its hash, and IP addresses and domain names can be easily changed. Moreover, such methods are entirely unable to detect *zero-day* attacks. The seemingly interminable rise in the number of successful network intrusions clearly demonstrates that signatures alone are not enough.

Hero et al. (2023) provide an overview of the current problems and challenges arising from cybersecurity threats in large enterprise systems, and the role of statistical and data science methods to address them. The promise of statistical approaches lies in anomaly detection capabilities, which can be used to complement existing cyber-defence systems and enhance the abilities of cyber security professionals. Commonly, statistical and machine learning methods are used to characterise the normal behaviour of some aspect of

the network and subsequently to detect anomalous departures from normal behaviour (Bhuyan et al., 2014; Mukherjee et al., 1994; Ye et al., 2002). In this way, *anomaly detection* does not explicitly identify malicious behaviour, but rather locates unusual and surprising activity, which may be symptomatic of an intrusion. The outputs of collections of anomaly detectors are used by security analysts to determine where to focus attention in seeking to uncover malicious behaviour. This toolkit aims to capitalise on the vast quantities of data that can be collected relating to network activity.

At first glance, it is surprising that the *deployment* of machine learning techniques for cyber security in operational environments has been relatively limited when compared to other data-rich environments such as image recognition, speech recognition and recommender systems. In Sanna Passino et al. (2023a); Sommer and Paxson (2010), a number of characteristics specific to the cyber security domain are identified, which present challenges to the successful implementation of machine learning. Some of these challenges relate to the sheer volume and velocity of data. Another challenge is a lack of training data—or more specifically, a lack of labels. Due to privacy and security concerns, and the prohibitively high cost of manually labelling data, real-world computer network data with *ground truth* malicious behaviour is extremely hard to come by. This dearth of labelled data precludes supervised learning methods and makes performance evaluation difficult for unsupervised methods. We propose a strategy for addressing this challenge in Section 3.2.

A further challenge relates to *false positives*—benign events that are flagged as anomalous. As the number of events corresponding to real attacks is typically very small in relation to the vast quantity of data analysed, even a low false positive rate can produce a very large number of false alarms. Anomaly detectors in this context are particularly prone to false positives due to the difficulty of defining "normal behaviour" in a complex, chaotic system such as an enterprise computer network (Michailidis, 2023). Anomalous behaviour is not always malicious; likewise, malicious behaviour may not appear anomalous. By combining data from multiple sources, we hope to reduce the model's susceptibility to false positives (i.e. increase precision)

while also increasing its recall. Consider a motivating example: to carry out an attack, a malicious agent typically will have to traverse across a number of devices on the network, authenticate credentials, run processes etc. These elements of behaviour can be separately observed from Netflow records, authentication logs and process logs, respectively. It may happen that certain activity of each type may not appear anomalous when viewed in isolation but their *co-occurrence* is highly anomalous. Conversely, an event that may appear anomalous in one view may be explained by co-occurring events in other views.

3.2 Supervised Learning using Time-shifted Labels

As a first test for predictive signals within cyber security data sources, we propose a method for contriving labelled training data that can be used in supervised learning models. We consider defining *data-determined* labels based on "what happens next?"; that is, we can monitor a quantity of interest in contiguous time intervals, or *bins*, within the data and define the label for bin t as the quantity observed in bin $t + 1$. In this way, it may be possible to learn signals that predict future activity.

It is critical that statistical approaches be guided by security expertise; the activity being modelled must be relevant to the cyber security problem. It is commonly known that *novelty* is often of direct concern to security analysts. For this reason, the future behaviour we consider is 'new events' occurring on a computer network. For the purpose of this study, we define a new event to be a computer communicating for the first time with a device with which it has not previously communicated. If these new events can be predicted from the data, then a security analyst could focus attention on those new events that occur despite a low predicted probability of occurrence. Novelty could be defined to refer to many other quantities (e.g. new server ports being accessed, new users on a host) and the study which follows adapted appropriately. Indeed, by simultaneously monitoring a collection of quantities, a clearer picture of anomalous activity can be formed.

3.2.1 Time-shifted Label

Rather than examining events in isolation, it is common in cyber security to group together events occurring within fixed intervals of time to better represent the behaviour of a given computer or of the network as a whole at that time. The choice of bin length will depend on a number of factors, such as the time resolution of the available data, the occurrence frequency of the event of interest and the computational demands of the model being employed. Interval lengths employed in related studies include 1 minute (Nezhad et al., 2016), 5 minutes (Whitehouse et al., 2016), 15 minutes (Riddle-Workman et al., 2018) and 30 minutes (Neil et al., 2013). For our experiments, we choose to divide each day's data into disjoint 5-minute bins.

The label we define for this study refers to the presence of a network communication, as featured in the Netflow subset of the LANL dataset. The Netflow data for each time bin t can be considered as a graph containing nodes C_i and edges $a_{i,j}^{(t)} \in \{1, 0\}$ indicating the presence/absence of a communication from C_i to C_j in bin t. For computer C_i we observe $D_{i,t} = \{C_j \ ; \ a_{i,j}^{(t)} = 1\}$, the set of devices with which it communicates in bin t. We wish to predict if $D_{i,t+1}$ contains previously unseen devices.

By introducing a time-shifted label for each computer, this can be treated as a binary classification problem. We define a label $y_{i,t} \in \{0, 1\}$, which indicates whether C_i makes a novel communication in bin $t + 1$:

$$y_{i,t} = \begin{cases} 0 & \text{if } D_{i,t+1} \subseteq D_{i,1:t} \\ 1 & \text{otherwise,} \end{cases}$$

where $D_{i,1:t} = D_{i,1} \cup \ldots \cup D_{i,t}$ is the set of all computers C_i has communicated with up to and including bin t.

3.2.2 Data Fusion

From each of our two datasets—Netflow and WLS—we can construct features relating to each computer's activity in bin t, to serve as input to a

supervised learning model. The LANL data is full of structure, and there are many ways in which features could be defined or engineered. The Netflow dataset is inherently a directed graph, with nodes representing devices and edges representing communications, which could even be weighted according to the number of packets/bytes sent and received. Alternatively, each computer's total inbound and outbound communications during a bin could be represented as a time series of byte values. Meanwhile, the process events from the WLS dataset can be represented as a tree structure, while the authentication events can be viewed as a bipartite graph between users and hosts. It is easy to see that modelling the LANL enterprise network data is a multi-view learning problem; beyond even these four highly heterogeneous feature representations just mentioned, many more are conceivable.

However, as discussed in Section 2.2, in order to combine signals across heterogeneous feature representations via early fusion, we are typically forced to convert each view from its natural structural representation into a numeric vector so that these can be concatenated. In Table 3.1, we present naïve numeric feature vectors constructed to summarise the activity in each dataset within a time bin. Clearly, a lot of potentially valuable information may be lost in this process.

For single-view, or late fusion multi-view learning, bespoke models can be fit for each different feature representation. While we expect enhanced predictive power could be gained by tailoring models to the more structured feature representations described above, we restrict this study to using the feature vectors given in Table 3.1, even for single-view and late fusion experiments. The aim of this study is simply to investigate whether statistical signals exist that can be used to predict network communications—and in particular, we are interested to see whether activity in the intra-device (WLS) dataset is predictive of activity in the inter-device (Netflow) dataset, and whether any additional predictive power can be gained by combining signals from both datasets. In later chapters, we develop models that can combine the benefits of both early and late data fusion approaches; that is, taking advantage of each view's natural structural representation, while also capturing dependence between views.

WLS		Netflow	
Name	Description	Name	Description
Time	Bin no. (1-288)	*Time*	Bin no. (1-288)
WorkHours	Working hours indicator	*WorkHours*	Working hours indicator
Events	No. of events	*OutDegree*	No. of unique outward communications
UniqueEvents	No. of unique event types	*InDegree*	No. of unique inward communications
Logons	No. of successful log-ons	*TotalOut*	Total no. of outward communications
FailedLogons	No. of failed log-ons	*TotalIn*	Total no. of inward communications
Users	No. of users active	*PacketsOut*	Total no. of packets sent
Processes	No. of processes started	*PacketsIn*	Total no. of packets received
		BytesOut	Total no. of bytes sent
		BytesIn	Total no. of bytes received
		UPD	Total no. of UDP flows
		NewEvents	No. of new events in the current bin

Table 3.1: Features extracted from each LANL dataset.

3.2.3 Entity Fusion

We expect there to be some correlation between data arising not only from the two data sources but also from different individual devices, or *entities*. A classifier trained using data from a group of computers may reveal relationships between the features that a classifier trained on any single computer fails to identify. Combining information from multiple computers may also mitigate the problem of overfitting the classifier to an individual computer's training data.

3.3 Experiment

A random selection of 500 computers was selected from the LANL WLS dataset. Each of these computers is also present in the Netflow dataset, ensuring that feature vectors can be constructed from both datasets. For each C_i, we use the first day of data to *burn in* $D_{i,1:288}$, i.e. we populate a list of all the computers with which C_i has communicated. This allows us to label data for subsequent time bins, continuously updating $D_{i,1:t}$. We use day-2 data, $t \in [289, 576]$, as a training set and day-3 data, $t \in [577, 864]$, as a test set.

For this experiment, we use a random forest classifier (Breiman, 2001), as

it is often regarded as being among the best 'out-of-the-box' classifiers; i.e. minimal tuning or pre-processing is required. For each computer separately, we train four classifiers: early fusion, late fusion, and single-view classifiers from the WLS and Netflow features. For early fusion, the feature sets from the two views (Table 3.1) are merged to form a fused feature set. For late fusion, separate classifiers are built using the WLS and Netflow feature sets, and the output class predictions are then averaged to determine the fused prediction.

In order to assess and compare the performance of the different families of classifier, we plot receiver operating characteristic (ROC) curves. Note that for each classifier type, we have a set of predictions from 500 separately trained models. This calls for special consideration regarding how to combine these predictions into a single ROC curve per classifier that faithfully represents its overall performance. This topic will be explored in detail in Chapter 4. In Hogan and Adams (2018), the predictions made by each model are simply pooled together per classifier (see Section 4.3.1), and ROC curves are constructed in the usual way. This is shown in Figure 3.1. To assess overall performance, we compute the AUC (area under the curve). The performance of each is remarkably similar. Of particular interest is the performance of the WLS classifier—there appears to be almost as much signal for predicting a new Netflow event in the WLS features as there is in the Netflow features. Early fusion of the two sets of features appears to result in a marginal improvement, suggesting informative interactions between the views may exist.

As discussed in Section 3.2.3, we expect it may be possible to improve predictive accuracy by combining training data among entities. We could use hierarchical clustering on the feature vectors of each computer to identify groups of 'similar' computers, for whom the fusion of training data may improve classification performance. However, these feature vectors are expected to contain a lot of noise. The random forest classifier extracts from these vectors the signal that predicts new edges for a given computer, and it is the similarity of the signals across computers that is of interest, rather than the similarity of the raw data, which contains both signal and noise. We can

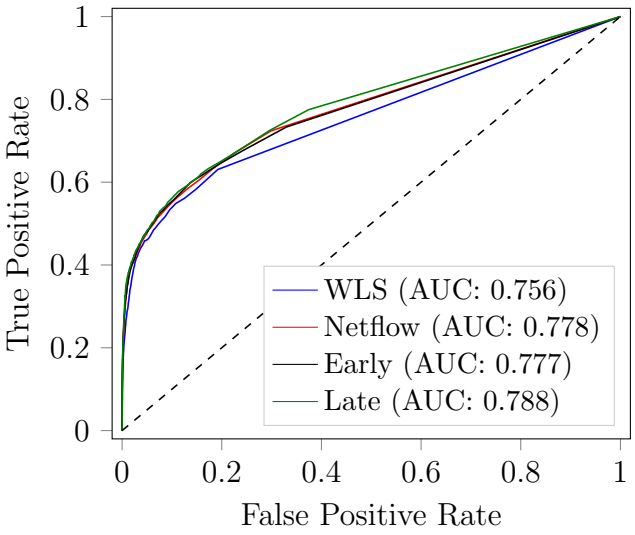


Figure 3.1: Pooled receiver operating characteristic curves for single-view and multi-view classifiers trained on 500 computers from the LANL dataset.

instead carry out hierarchical clustering on the vectors of importance scores assigned to each feature, which is based on the Gini impurity index (Breiman et al., 1984, p. 103) used for the calculation of splits during training. Doing so reveals two well separated clusters of roughly equal size. We note that for one cluster, the *Time* feature is given appreciably more importance.

New training sets are constructed by merging the individual training sets of each computer in a cluster. We then carry out the same tests as before and plot ROC curves (Figure 3.2). Interestingly, classification is considerably more accurate for cluster one than for cluster two. Also, the early fusion classifier performs the best for cluster one, indicating that informative relationships between the datasets are revealed when data from several computers is combined. None of the classifiers performs particularly well in cluster two. The fact that the classifiers for these computers assigned a relatively higher importance to the *Time* variable suggests that there is less signal in the data from these computers.

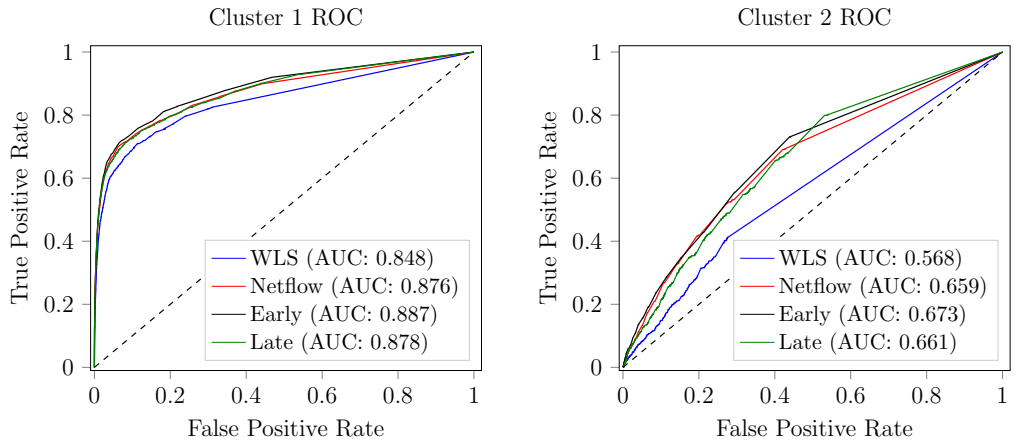


Figure 3.2: ROC curves of classifiers trained using combined data from clusters of computers.

3.4 Discussion

We have described how statistical modelling of enterprise network data represents a paradigmatic multi-view learning problem. Using a real-world public-domain dataset, we illustrated the inherently heterogeneous, multi-view nature of enterprise network data and described some of the challenges that arise when trying to combine information from views with diverse underlying representations. A further challenge is presented by a dearth of labels. We demonstrated a framework for contriving labelled training data for supervised learning and used this to investigate the predictability of Netflow events. The results are highly promising and suggest that signals exist in both views. Considering the level of information loss incurred by constructing the rather crude features of Table 3.1, we are motivated to develop more sophisticated techniques that can appropriately handle data from heterogeneous views, while capturing dependence between views. This will be the focus of Chapters 5 to 7. First, we discuss an issue that arises when combining the outputs of multiple classifiers into a single ROC curve, as was done in Figure 3.1.

4

On Averaging ROC curves

Receiver operating characteristic curves are a widely used method of presenting results from classification studies. The ROC curve describes the separability of the distributions of predictions from a *single* two-class classifier. However, in the previous chapter, for each learning scheme we constructed an ROC curve to assess the performance of *multiple* (500) individually trained models. More generally, there are a variety of situations in which an analyst seeks to aggregate the predictions made by multiple classifiers into a single representative example. A number of methods of doing so are available; however, there is a degree of subtlety that is often overlooked when selecting the appropriate one. An important component of this relates to the interpretation of the decision process for which the classifier will be used. In this chapter, we take a momentary detour to summarise a number of methods of aggregation and carefully delineate the interpretations of each in order to inform their correct usage. Through a toy example, we highlight how an injudicious choice of aggregation method can lead to erroneous conclusions and misrepresentation of results. The content of this chapter is taken from Hogan and Adams (2023).

The remainder of the chapter proceeds as follows: Section 4.1 introduces

the situations in which average ROC curves are used and discusses relevant literature. Section 4.2 provides the required definitions and notation for individual ROC graphs. In Section 4.3 we describe and illustrate the construction of average ROC curves under the three proposed combination methods. In Section 4.4, a discussion of the interpretations of different aggregating methods motivates guidelines for selecting the appropriate method. Finally, Section 4.5 provides a simple illustration to demonstrate that the choice of combination method is crucial in the context of comparing classifiers.

4.1 Background and Relevant Literature

ROC curves are widely used to present performance assessment results in classification studies. Their usage spans many disciplines, including pattern recognition (Webb, 2003), medical diagnostics (Swets et al., 2000), consumer credit scoring (Hand and Henley, 1997), and biometrics (Ross and Jain, 2003). Typically, the design and construction of a classifier is an iterative process: at each stage, choices are made regarding feature selection, distributional assumptions etc., and the classifier must be re-evaluated to assess the impact of these choices on performance. In many classification problems, such as those with imbalanced classes or asymmetric misclassification costs, a scalar measure of performance such as accuracy is not sufficient (Provost and Fawcett, 1997). In addition, it is often the case that the misclassification costs and class frequencies are not known at experimentation time and will only become clear at the time of deployment (Drummond and Holte, 2000; Hernández-Orallo et al., 2012). ROC curves offer a useful visual evaluation of the trade-off between different types of error over the full range of possible costs and class frequencies. By examining a ROC curve, various operating points of interest—motivated by the specific application of the researcher—can be compared. If the operating conditions at deployment time are known precisely, then a plot of the entire ROC curve is redundant, as algorithms can be compared directly based on specific cost-sensitive classification metrics.

There are many situations in which there is a requirement to produce a *single* summary ROC curve from a collection of individual curves. For example,

in credit card fraud detection, a bank may be running classifiers based on the same features for each customer (Juszczak et al., 2008); in biometric face verification, separate classifiers are trained for each individual subject (Marcialis and Roli, 2002); in radiology, multi-reader multi-case (MRMC) studies involve multiple radiologists producing scores for a collection of image cases (Skaron et al., 2012). Another situation involves the use of k-fold cross-validation procedures, which attempt to provide some sense of the average and the uncertainty in the ROC curve. It is perhaps natural in these settings to seek to report a single ROC curve summarising the performance of all the individual classifiers. Such a curve will be useful for classifier development and selection. Note that a related problem is that of constructing a *covariate-adjusted* ROC curve (Janes and Pepe, 2009), which aims to account for factors or characteristics that influence the predictions made by a classifier on certain instances within a single dataset. The focus of this paper is on constructing a single ROC curve from *multiple* datasets.

Reasoning about a summary or "average" ROC curve calls for fundamental considerations relating to the operational use of the classifier. Crucially, an aggregate ROC curve should represent performance in accordance with how the individual classifiers are to be deployed. As described in Section 4.2, a threshold T is required to produce a decision from most classifiers. In this context featuring multiple classifiers, the question is whether every classifier is to employ the same threshold T or if classifier i operates using threshold T_i. Next, it is important that the method of aggregating individual ROC curves is compatible with the application-specific operating points or characteristics of interest, upon which performance assessments will be made.

The literature contains a number of methods for aggregating ROC curves (Fawcett, 2006; Swets and Pickett, 1982); however, clear guidance on their correct usage is lacking. Discussing two methods, *vertical averaging* and *pooling* (see Section 4.3), Witten et al. (2011) remark that "it is not clear which method is preferable. However, the latter method is easier to implement." Similarly, Parker et al. (2007) comment that "both strategies are used in practice and the current literature is equivocal about which approach is to be recommended." This is also clear when one considers software packages for

ROC analysis; the R package `ROCR` offers three methods for averaging ROC curves, though neither its documentation (Sing et al., 2015) nor accompanying paper (Sing et al., 2005) provide any guidance on which is appropriate for different situations. An online vignette by the authors[1] demonstrates *threshold averaging* for combining the results of cross-validation. Meanwhile, a similar tutorial[2] for the Python package `scikit-learn` (Pedregosa et al., 2011) demonstrates *vertical averaging* in the same context.

This chapter aims to address this lack of guidance; we will show that care must be taken when choosing the appropriate averaging method, in particular with respect to the fundamental considerations described above. We will provide an example that demonstrates that incorrectly combining curves can lead to misrepresentation of results. The purpose of this chapter is not to provide a deeply novel result but rather to clearly delineate the characteristics of different aggregation procedures and provide guidance for their correct usage.

4.2 ROC Graphs

Here we follow closely the framework presented by Krzanowski and Hand (2009). Suppose that there exists two populations—a 'positive' population P and a 'negative' population N—together with a classification rule for allocating unlabelled instances to one or other of these populations. A classification rule (*classifier*) is a function $S(X)$ of the random vector X of variables measured on each instance, used to predict class membership of a given instance. Some classifiers produce as output a discrete class label indicating only the predicted class of the instance; others produce a continuous output or *score*. The focus of this chapter is the latter set of classifiers, as it is only for these that ROC curves can be produced. Suppose $\boldsymbol{x}$ is the observed value of X for a particular instance. The score $s(\boldsymbol{x})$ can be used to predict class membership according to whether $s(\boldsymbol{x})$ exceeds or does not exceed some threshold

[1] `https://ipa-tys.github.io/ROCR/articles/ROCR.html`

[2] `https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html`

T. As S is a continuous variable, we can consider the distribution of scores pertaining to instances from populations P and N. Let $f_{S|P}(s) = p(s \mid P)$ and $f_{S|N}(s) = p(s \mid N)$ be the probability density functions of scores in P and N, respectively, with corresponding distribution functions $F_{S|P}$ and $F_{S|N}$. A ROC graph is essentially a visual description of the degree of separability of these score distributions generated by a particular classifier.

We do not discuss how a classifier should be chosen or indeed calibrated for a particular task. These are well studied problems (Vapnik, 2013; Wolpert, 1996). Rather, we discuss how ROC curves are used to assess the performance of classifiers, in particular when applied to multiple datasets.

4.2.1 Definition

The score produced by a classifier can be a probability, representing the likelihood that an instance is a member of a particular class (the positive class, by convention), or simply a general, uncalibrated score, in which case it is conventional that higher scores are more indicative of positive class membership and lower scores indicative of negative class membership. In either case, the choice of threshold T will dictate the class assignments made by the classifier. Suppose that t is the value of the threshold T chosen for a particular classifier; an instance is assigned to P if its score s exceeds t, otherwise N. In order to assess the efficacy of this choice of T, we must consider four possible outcomes and the *rate* at which these occur:

1. an instance from P is correctly classified, i.e. the true positive rate $tp = p(s > t \mid P)$;
2. an instance from N is misclassified, i.e. the false positive rate $fp = p(s > t \mid N)$;
3. an instance from N is correctly classified, i.e. the true negative rate $tn = p(s \leq t \mid N)$;
4. an instance from P is misclassified, i.e. the false negative rate $fn = p(s \leq t \mid P)$.

The choice of operating threshold t is application-specific and often complicated. By varying t and evaluating the four quantities above, we can inform a decision regarding which value to choose. In fact, as $tp + fn = 1$ and $fp + tn = 1$, we need only consider two of the above quantities, typically fp and tp. A ROC curve is obtained by varying t from $-\infty$ to ∞ and tracing a curve of (fp, tp). We can write down the equation of this curve in terms of the distribution functions defined above:

$$\begin{aligned} tp &= p(s > t \mid P), \quad -\infty < t < \infty \\ &= 1 - F_{S|P}(t), \quad -\infty < t < \infty \\ &= 1 - F_{S|P}\left[F_{S|N}^{-1}(1 - fp)\right], \quad 0 \leq fp \leq 1. \end{aligned}$$

4.2.2 Estimation

The role of the ROC curve is to present an assessment of the performance of a classifier over the whole range of potential thresholds. This performance will be determined by the degree of overlap of the scores assigned by the classifier to instances from P and N. In practice, we hardly ever know anything about the true underlying distributions of these scores. One typically only has available the values of X for a set of instances whose class labels are known. The data is commonly split into two portions: a *training set* is used to estimate any parameters required by the chosen classifier, and this classifier can then be applied to a *test set* in order to estimate the score distributions, and hence fp and tp.

Standard methods of statistical inference are available to the researcher for this task. One may wish to assume parametric models for $F_{S|P}$ and $F_{S|N}$ and estimate parameters using maximum likelihood on the sample data. However, in this case, caution must be taken as the accuracy of the resulting ROC curve and any derived quantities strongly depends on the validity of the assumptions made (Krzanowski and Hand, 2009; Zhou et al., 2009). If a large number of test instances are available, then empirical estimation is preferred.

Let n_P and n_N be the number of instances in the samples from populations P and N, respectively. We write $n_{P(t)}$ and $n_{N(t)}$ for the number of those instances whose classification scores are greater than t. Then the empirical estimators of tp and fp at the classifier threshold t are given by

$$\widehat{tp} = \frac{n_{P(t)}}{n_P} \qquad \text{and} \qquad \widehat{fp} = \frac{n_{N(t)}}{n_N}.$$

The empirical ROC curve is then constructed by plotting the points $(\widehat{fp}, \widehat{tp})$ for varying t.

4.2.3 Area Under the ROC Curve

A quantity widely used to summarise an important element of the information portrayed by a ROC curve is the area under the curve (AUC). The AUC can be interpreted as an average true positive rate, regarding all values of the false positive rate as equally likely (Hand, 2009). Alternatively, it can be seen as the probability that the classifier will allocate a higher score to a randomly chosen instance from P than it will to a randomly and independently chosen instance from N. Clearly, a higher AUC is desirable; however, classifiers should not be compared solely on their AUCs, as a higher AUC does not imply an everywhere dominating ROC curve. As a scalar measure of performance, a lot of valuable information contained in the ROC curve is lost, such as performance in specific regions of ROC space that may be of interest to the researcher. Moreover, there exists a fundamental incoherence in the use of AUC to compare *different* classifiers—see Hand (2009) for details, and Ferri et al. (2011) for an alternative interpretation of AUC.

4.3 Averaging ROC Curves

The ROC graph described in the preceding section relates to a single dataset comprising instances belonging to either of two classes. Often, as in Section 3.3, we have multiple independent datasets of this type and a set of classification scores for each. In these settings, a separate ROC curve can be

constructed from each set of scores. One approach for comparing different classifiers, or different iterations during classifier design and development, would be to compare average AUCs. Equally, we could compare the average accuracy at some known operating point. However, as discussed in Section 4.1, it is often the case that full knowledge of the operating conditions at deployment time is not available during experimentation. This is precisely why ROC curves are useful, and hence why we may want to construct a single ROC curve that serves as a summary or average of the collection of individual ROC curves. In this way, the average performance at specific operating points can be assessed.

Methods for combining or averaging ROC curves have been proposed; however the properties of the resulting curves have not been properly explored nor compared in this context. Moreover, there is very little guidance in the literature regarding which of the methods is most appropriate for a given task; in fact, existing guidance can even be misleading. As we will see, the interpretation of a summary ROC curve differs according to the averaging technique used, and so care should be taken both in choosing the appropriate method and in presenting results. Note that methods for averaging parametric ROC curves have also been proposed, based on averaging the estimated parameters of each curve (Metz, 1989). However, the focus of this discussion is on empirical ROC curves.

In this section we adapt the notation slightly to allow for the case where there is a collection of datasets, each consisting of instances to be classified. Say that there are M datasets, and dataset i comprises a positive population P_i and a negative population N_i. A classifier produces scores for each instance, so the ROC curve for dataset i describes the separability of the score distributions $p(s \mid P_i)$ and $p(s \mid N_i)$.

For a simple illustration of the proposed averaging methods, we simulate scores arising from a probabilistic classifier employed on *two* datasets: the first 'well separable', the second 'poorly separable'. We assume equally balanced classes for both datasets but allow the total number of instances to differ between datasets. The number of scores simulated for the negative and positive classes of the well separable dataset was $n_{N_1} = n_{P_1} = 150$; for the

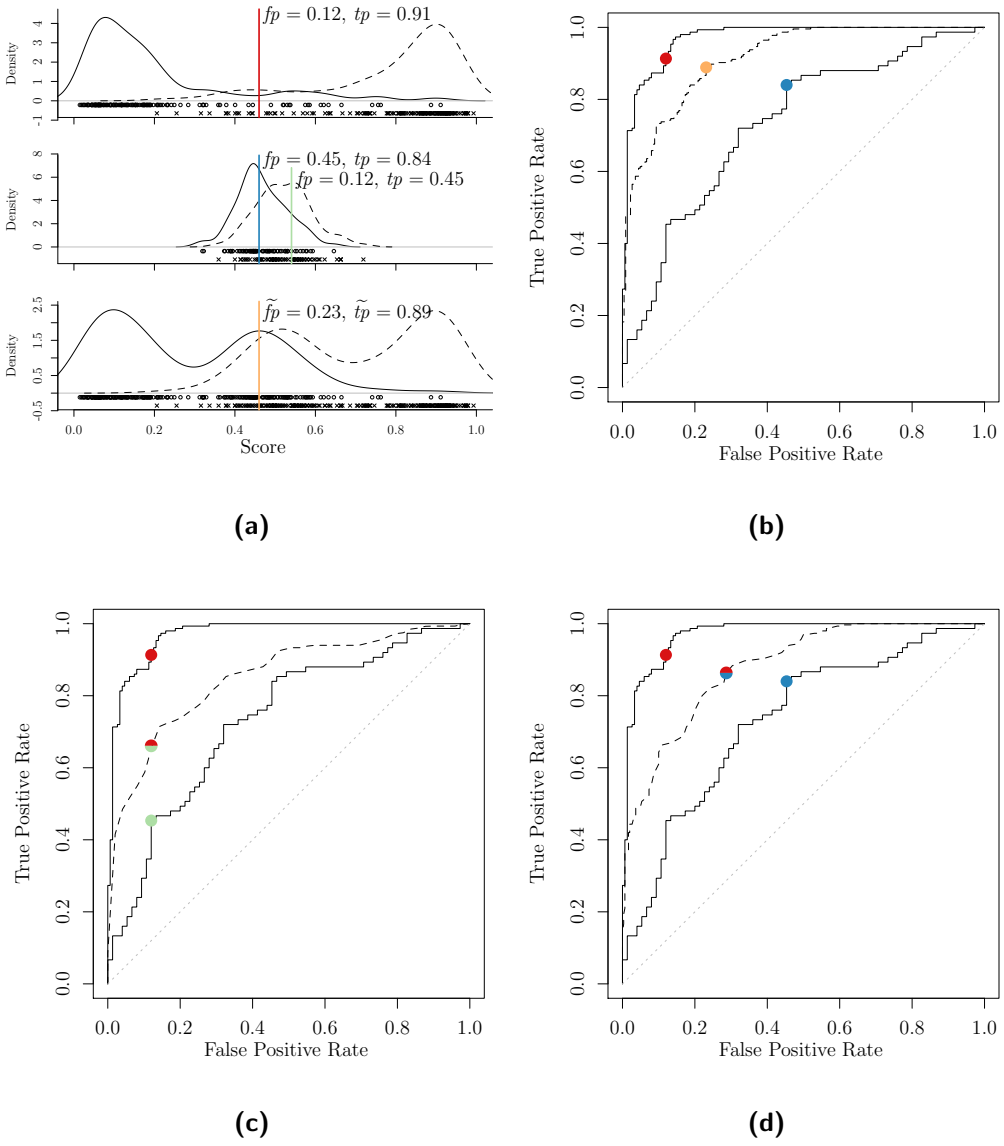


Figure 4.1: Illustration of three methods of averaging ROC curves. (a) Simulated classification scores for two entities (top and middle) with density estimates of the negative and positive score distributions. The bottom plot shows density estimates of the pooled score distributions. (b) ROC curve pooling. (c) Vertical averaging. (d) Threshold averaging. In (b)–(d), coloured markers correspond to the fixed thresholds in (a); a split-colour marker indicates an average of the rates at the corresponding thresholds.

poorly separable dataset $n_{N_2} = n_{P_2} = 75$. Density estimates of the simulated positive and negative scores for the two datasets are shown in the top two panels of Figure 4.1a.

4.3.1 Pooling

Swets and Pickett (1982) propose simply merging or *pooling* all the scores assigned to all instances from all datasets. A ROC curve is then constructed in the usual way as if every instance came from the one dataset:

$$\widetilde{tp} = \frac{\sum_{i=1}^{M} n_{P_i(t)}}{\sum_{i=1}^{M} n_{P_i}} = \frac{n_{P(t)}}{n_P} \qquad \widetilde{fp} = \frac{\sum_{i=1}^{M} n_{N_i(t)}}{\sum_{i=1}^{M} n_{N_i}} = \frac{n_{N(t)}}{n_N},$$

where P and N are the total number of positive and negative instances among all M datasets, respectively.

The pooled ROC curve is a representation of the degree of separability of the mixture distributions that result from pooling scores between the two datasets. Density estimates of these mixture distributions can be seen in the bottom panel of Figure 4.1a. A fixed threshold is shown in all three plots of Figure 4.1a and the corresponding points in ROC space can be seen on the ROC curves in Figure 4.1b.

4.3.2 Vertical Averaging

In Provost et al. (1998), *vertical averaging* is proposed, whereby points are sampled uniformly along the fp axis and the corresponding tp values from each individual ROC curve are averaged. Each ROC curve is treated as a function R_i such that $tp = R_i(fp)$. This is done by choosing the maximum tp for each fp sampled between 0 and 1, interpolating between points where necessary. The vertical average ROC curve is

$$\overline{R}(fp) = \frac{1}{M} \sum_{i=1}^{M} R_i(fp), \quad 0 \leq fp \leq 1.$$

Standard errors can also be calculated and used to construct confidence intervals for the average curve (a related but different problem is that of constructing confidence *bands* for a ROC curve estimated from a single dataset—see

Macskassy and Provost (2004)).

Note that the threshold that yields a given value of *fp* on each ROC curve will usually be different. The fixed threshold in Figure 4.1a yields a false positive rate of 0.12 for the well-separable dataset; the threshold yielding the same false positive rate for the poorly-separable dataset is shown in green. The corresponding true positive values for each dataset are shown on the ROC curves in Figure 4.1c, along with the vertical average curve.

4.3.3 Threshold Averaging

Instead of sampling points based on their positions in ROC space, *threshold averaging* (Fawcett, 2006) samples uniformly from the threshold values t and averages separately the *fp* and *tp* rates achieved by each ROC curve for that value of t.

$$\overline{tp} = \frac{1}{M}\sum_{i=1}^{M} tp_i \qquad \overline{fp} = \frac{1}{M}\sum_{i=1}^{M} fp_i$$
$$= \frac{1}{M}\sum_{i=1}^{M} \frac{n_{P_i}(t)}{n_{P_i}} \qquad = \frac{1}{M}\sum_{i=1}^{M} \frac{n_{N_i}(t)}{n_{N_i}}.$$

The threshold average ROC curve is shown in Figure 4.1d. Note that for the fixed threshold shown in Figure 4.1a, the average *fp* and *tp* values do not equal $\widetilde{fp}$ and $\widetilde{tp}$ (the pooled rates). As $n_{N_1} > n_{N_2}$ and $n_{P_1} > n_{P_2}$, the well separable scores dominate in the calculation of $\widetilde{fp}$ and $\widetilde{tp}$, so the pooled ROC curve is drawn closer to the well separable ROC curve. If instead $n_{N_1} = n_{N_2}$ and $n_{P_1} = n_{P_2}$, then the threshold average curve and the pooled curve would in fact be identical.

4.4 Interpreting Average ROC Curves

In most cases, the methods described each result in different average ROC curves. This can be seen clearly in the simple illustration provided in Figure 4.1. Furthermore, the interpretation of each is subtly different. It is important therefore to consider which method is most appropriate for a given

study.

Pooling classification scores disregards datasets. The pooled ROC curve can be considered as a weighted average of the individual ROC curves, weighted by the number of instances scored in each. Pooling produces only a single curve and so provides no information regarding uncertainty. If one dataset contains a substantially larger number of instances than the other datasets, the pooled ROC curve will be biased towards the ROC curve for that dataset. This should not be an issue in k-fold cross-validation, as the test sets are typically the same size. However, the pooling strategy—and equivalently, *threshold averaging*—assumes that the classifier outputs across folds of cross-validation are comparable and thus can be globally ordered. Parker et al. (2007) show that this assumption is generally not valid and can lead to large pessimistic biases in the estimation of the AUC and the shape of the ROC curve. Such biases are not suffered by *vertical averaging* (or its variants, discussed in Section 4.4.1); Chen and Samuelson (2014) prove that the AUC of the average curve equals the average of the individual AUCs.

Vertical averaging offers a way of obtaining such a measure of variance. In Fawcett (2006, p. 869), it is claimed that this method of averaging "is appropriate when the [false positive] rate can indeed be fixed by the researcher, or when a single-dimensional measure of variation is desired." *Threshold averaging* is presented as an alternative approach for situations when the false positive rate is not under the direct control of the researcher. The proposed solution is to average ROC points with respect to fixed thresholds, as these can be controlled by the researcher. We argue that this guidance is highly misleading, and the choice of averaging technique should instead be made under careful consideration of the *interpretation* of the resulting ROC curve. The vertical average ROC curve should be interpreted as the average true positive rate achieved amongst datasets for each fixed false positive rate, *allowing the threshold yielding that false positive rate to vary between datasets.* The threshold average ROC curve's interpretation is the average performance achieved amongst datasets *if the threshold is fixed across all datasets.*

The way in which the classifier is to be deployed should therefore be the first consideration to guide whether ROC curves should be averaged by threshold

or averaged in ROC space. If a fixed threshold is used in practice, then threshold averaging is appropriate. Otherwise, averaging should take place in ROC space and the researcher must next consider the characteristics of the curve upon which evaluation or comparisons will be made.

4.4.1 ROC Space Averaging

In situations where we do not assume a fixed threshold, vertical averaging is preferable to threshold averaging to provide a representation of the average performance achieved. Indeed, in many operational settings, the researcher is concerned with the the average true positive rate achieved for small false positive rates. However, in other settings it may be more relevant to assess the average false positive rate suffered in order to achieve a high true positive rate, say. This information will not be conveyed by the vertical average curve. Similarly, it is common in biometrics studies (Wayman, 1999) to consider a ROC curve in terms of the ratio of the false acceptance rate (fp) and the false rejection rate ($1 - tp$). The equal error rate (EER), the point at which these rates are equal, is then often used to assess and compare classifier performance. The vertical average ROC curve will again be a misleading description of the average performance when considered from this point of reference.

When averaging ROC curves in ROC space, the researcher must consider how the curve will be read, i.e. the axis of reference from which characteristics of interest are compared. To compare performance with respect to fixed false positive rates, averaging should take place *vertically*; for performance with respect to fixed true positive rates, averaging should take place *horizontally*; for performance with respect to fixed error rate ratios, averaging should take place *diagonally*.

All three approaches can be generalised into a single procedure for performing averaging in ROC space (Chen and Samuelson, 2014). The fp and tp axes

are first rotated clockwise by an angle θ by applying a rotation matrix

$$\begin{bmatrix} fp' \\ tp' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} fp \\ tp \end{bmatrix}.$$

The procedure is then the same as vertical averaging: values of fp' are sampled uniformly and the corresponding tp' values are averaged, yielding $\overline{tp'}$. The $(fp', \overline{tp'})$ pairs are then rotated anti-clockwise to the original ROC space:

$$\begin{bmatrix} fp \\ \overline{tp} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} fp' \\ \overline{tp'} \end{bmatrix}.$$

Vertical, horizontal and diagonal averaging can then be achieved using $\theta = 0$, $\pi/2$, $\pi/4$ respectively.

Averages along other directions may also be of interest. For example, denote the *cost* of misclassifying an instance from P and N by c_P and c_N, respectively. If the ratio of these costs is known or estimated, the threshold that minimises cost corresponds to the point of intersection of the ROC curve and a line originating from the point (0,1) with slope $-\pi_P c_P / \pi_N c_N$, where π_P and π_N are the probability that an instance comes from P and N, respectively (Adams and Hand, 1999). Hence, ROC curves can also be averaged with respect to fixed costs using $\theta = \arctan(\pi_N c_N / \pi_P c_P)$. However, Adams and Hand (1999) point out that it is rare in practice for these costs to be known precisely, hence the pre-eminence of ROC curves in practical applications.

4.5 Simulated Example

We now present an example of a realistic use-case of ROC curve averaging in which the resulting conclusion depends on the method of averaging used. Suppose a researcher wishes to compare the performance of two different classifiers on data arising from a collection of M datasets. Classification rules are learnt and scores generated by both classifiers using data from each dataset separately. The aim is to choose the classifier that has the better overall or average performance. We illustrate two scenarios.

Scenario 1. The output classification scores for two arbitrary classifiers were simulated as follows: For classifier 1 (C-1), negative and positive class scores for dataset i are simulated from Gaussian distributions $N(\mu_{i0}, \sigma_{i0})$ and $N(\mu_{i1}, \sigma_{i1})$ respectively. For classifier 2a (C-2a), the scores for dataset i are simulated from $N(\mu_{i0} + \varepsilon_i, \sigma_{i0})$ and $N(\mu_{i1} + \varepsilon_i, \sigma_{i1})$, where $\varepsilon_i \sim N(0, 1)$ is some random noise that differs between datasets but is the same for positive and negative populations within a dataset. As such, the locations of the modes of the score distributions of the two classifiers will be shifted but the class separability will be the same.

The first two panels of Figure 4.2 show ROC curves for both classifiers constructed using scores for $M = 100$ datasets simulated in this way. The vertical and threshold average ROC curves are shown for both classifiers. 95% confidence intervals for the averages were constructed at each point using the assumption of normality. Note that the vertical average curves remain approximately equal for both classifiers but the threshold average curves differ. This highlights the importance of considering the interpretations of average ROC curves in order to avoid erroneous comparisons. Here, if the operational intention is to fix a threshold across all datasets, but vertical averaging is used, no difference between the classifiers is evident and the researcher will choose either.

Scenario 2. The opposite scenario is also possible. We simulate scores for an alternative classifier (C-2b), this time shifting the modes of C-1's score distributions unequally in order to increase the class separability. We do this by simulating negative class scores for dataset i from $N(\mu_{i0} + \varepsilon_i, \sigma_{i0})$ as we did for C-2a but simulating positive class scores from $N(\mu_{i1} + |\varepsilon_i|, \sigma_{i1})$. For datasets where $\varepsilon_i < 0$, the score distributions become more separable. In Figure 4.2, the vertical average ROC curve for C-2b dominates both average curves for C-1; however the threshold average ROC curve remains approximately equal. Again, an averaging method unsuitable for the operational intentions can result in a misguided choice of classifier.

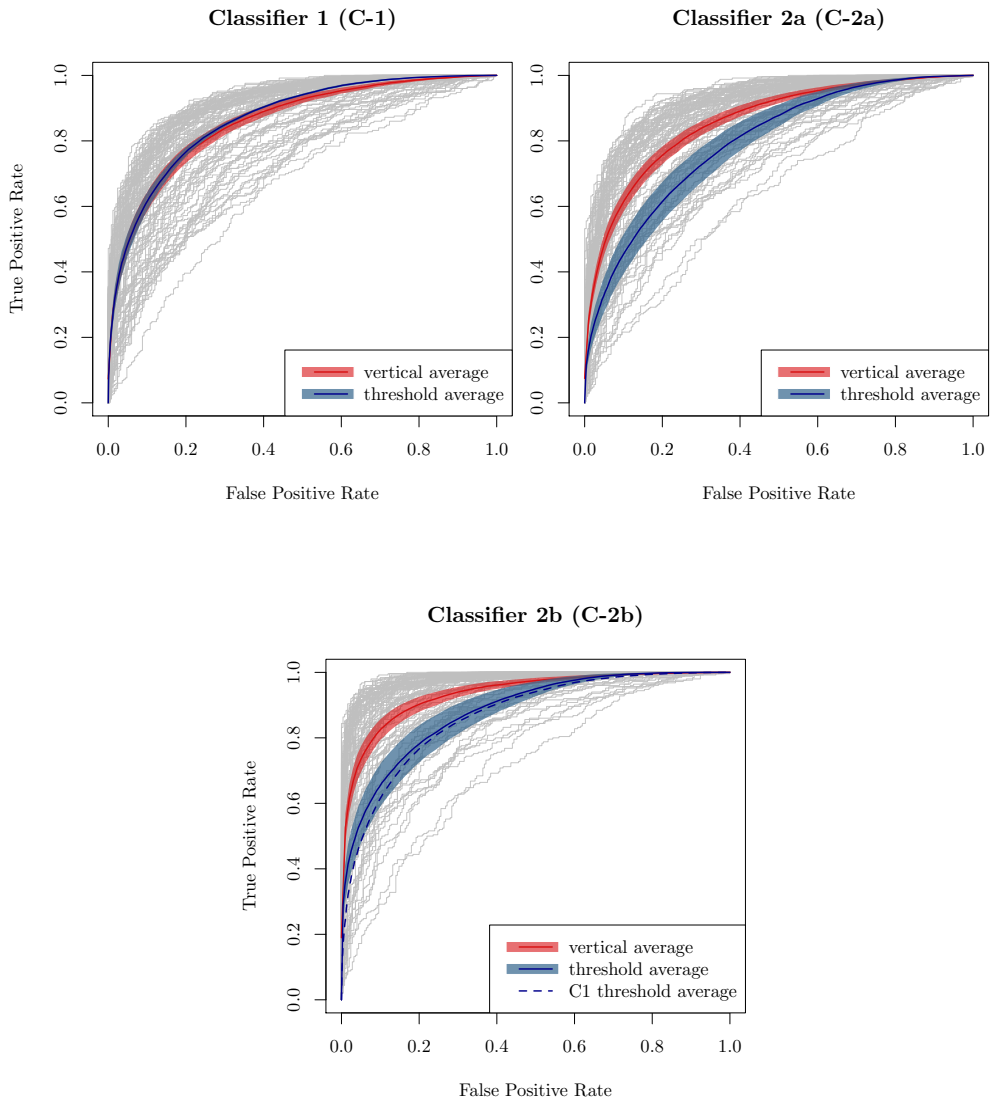


Figure 4.2: Simulated ROC curves for three classifiers and their averages under vertical and threshold averaging. The locations of the distributions used to simulate classification scores for C-1 are perturbed such that, for C-2a, the ROC curves—and hence the vertical average curves—are comparable but the threshold average curves differ, and for C-2b, the threshold average curves are comparable but the vertical average curves differ.

4.6 Discussion

This chapter was prompted by the requirement in Hogan and Adams (2018) to assess the aggregate performance of multiple separately trained classifiers. The prevalence of average ROC curves in the literature—often without reference to the averaging technique used—warranted a formal exploration of the topic. Here we have summarised the common methods and pointed towards their interpretation in the operational context. Fundamentally, the appropriate choice of combination method *must* correspond to the operational approach intended for the classifier. To drive this message home, we provided a simple illustration to show that incorrect decisions can result from an ill-considered choice of combination method. We hope to have highlighted that in the context of classifier design and development, valuable information may be lost if due care and consideration is not paid when presenting and assessing results. In the following chapters, we return our attention to multi-view learning.

5

Semi-supervised Learning via Co-regularised Kernel Logistic Regression

In Chapter 3, we highlighted the unavoidable trade-off that is presented by early and late data fusion strategies; that is, when heterogeneous feature representations are available, we are forced to compromise between prioritising the potential signal contained within each view's natural structural representation and the potential signal unlocked by combining information between views. In this chapter we explore the use of semi-supervised learning as a means of modelling a dependence between modalities of multi-view data. Such an approach is ideally suited to malware detection—an area of cyber security in which it is possible, but expensive, to obtain labelled training data, while additional unlabelled data is abundant. Motivated by this, we develop a probabilistic classifier that can learn from heterogeneous feature representations and simultaneously harness unlabelled data to capture their interdependence.

The remainder of the chapter is organised as follows: We begin in Section 5.1

with a review of background literature for semi-supervised learning. Additionally, we discuss how vector-valued kernel methods offer a powerful framework for tailoring models to multi-view data, which is often non-numeric. In Section 5.2, we introduce the concept of reproducing kernels and define the feature spaces in which they implicitly operate. For clarity, we illustrate the scalar case first and present the vector-valued case as a natural extension thereof. In Section 5.3, we describe an abstract multi-view semi-supervised learning framework, based on classical structural risk minimisation. This framework has formed the basis for multiple semi-supervised learning implementations in the literature. In Section 5.4, we propose a logistic regression implementation and describe in detail its optimisation procedure. In Section 5.5, we test our algorithm using synthetic data and subsequently perform a malware classification experiment using the EMBER dataset.

5.1 Background and Relevant Literature

When labelled training data is available, the aim of machine learning is to estimate some target function $f: \mathcal{X} \to \mathcal{Y}$, which maps elements of the input space $\mathcal{X}$ to responses from the output space $\mathcal{Y}$. With multi-view data, $\mathcal{X} = \mathcal{X}^{(1)} \times \ldots \times \mathcal{X}^{(v)}$, so there are multiple hypothesis spaces, each containing candidate functions. If we assume that each contains a function that well-approximates the target function, then, intuitively, if these functions agree with the target function, they should also agree with each other on unlabelled samples. By rejecting functions from separate hypothesis spaces that "disagree" with each other, the complexity of the joint learning problem is reduced.

This is the basis for the co-training algorithm of Blum and Mitchell (1998), which gave rise to the field of multi-view learning as a self-contained paradigm. Co-training is initialised by supervised classifiers in each of two views, which are then alternately re-trained using a training set augmented with the unlabelled samples most confidently predicted by the other classifier. Under the assumptions that there exist weak classifiers in each view and that the features from each view are conditionally independent given the class label,

Blum and Mitchell (1998) provided PAC-style guarantees that co-training can use unlabelled data to learn arbitrarily strong classifiers. A large body of literature followed, including extensions, applications and theoretical analyses. For example, the co-EM algorithm of Nigam and Ghani (2000) combines co-training and Expectation Maximisation (Dempster et al., 1977) to operate simultaneously on all unlabelled samples using classifiers that can learn from probabilistically labelled training sets. They also show that co-training on multiple views manually generated by random splits of features can result in performance improvements even when no natural partitioning of the features is apparent. SimCLR (Chen et al., 2020), a recent state-of-the-art semi-supervised learning method based on neural networks, manually generates multiple views of unlabelled data through augmentation (e.g. cropping, addition of Gaussian noise).

As we discussed in Section 2.2, for most interesting multi-view learning problems, the different views can be clearly partitioned based on their heterogeneous modalities, e.g. images, text, histograms and time series. The requirement to be able to handle diverse data structures immediately motivates the use of kernel methods for multi-view learning. Ever since seminal work by Boser et al. (1992) (although their history goes back as early as Aizerman 1964; Aronszajn 1950; Parzen 1962), kernel methods have remained among the most powerful paradigms in modern statistical machine learning. In addition to their ability to efficiently represent linear patterns in high dimensional space, a major strength of kernel methods—first noted by Schölkopf (1997)—is their ability to operate on symbolic objects, such as graphs, sets, strings, images, sequences or text documents. As long as a valid (see Section 5.2.1) kernel function can be constructed, any kernel method can be applied. Such a kernel function automatically implies a vectorial representation of the data in some reproducing kernel Hilbert Space (RKHS; see Definition 5.2.2).

Methods for two-view learning based on RKHSs, which can be solved by direct optimisation, were developed by Farquhar et al. (2006); Sindhwani and Rosenberg (2008); Sindhwani et al. (2005). These so-called *co-regularisation* methods optimise an objective function that includes a regularisation term that explicitly penalises disagreement between the predictions of the two

views. Recently, there has been increased research attention focused on RKHSs of *vector-valued* functions (Caponnetto et al., 2008; Carmeli et al., 2006; De Vito et al., 2013; Micchelli and Pontil, 2005; Zhang et al., 2012). These naturally extend kernel methods from single-output to multiple-output (or so-called *multi-task*) learning problems. Kadri et al. (2013) first recognised the opportunity to leverage this same framework for multi-view learning. By simply converting a scalar-valued response to a vector with equal components, the vector-valued regularised least squares framework of Evgeniou et al. (2005) can be readily applied to data with an arbitrary number of views. Extensions and improvements by Minh et al. (2013, 2016) and Huusari et al. (2019) followed, demonstrating the vector-valued RKHS framework to be highly apposite for the requirements of multi-view learning; that is, separate kernel functions can be tailored to the characteristics of each individual view, while also allowing between-view information to be incorporated into the model.

5.2 Reproducing Kernel Hilbert Spaces

In this section, we introduce the vector-valued reproducing kernel Hilbert space and its associated operator-valued kernel. We consider these subjects only to the extent needed to understand the multi-view regularisation framework. For a more detailed discussion, see e.g. Caponnetto et al. (2008); Carmeli et al. (2006); Evgeniou et al. (2005); Micchelli and Pontil (2005).

5.2.1 Scalar-valued Kernel Machines

The idea of kernel machines is to learn a linear function or decision boundary in some high-dimensional space that corresponds to the image of a non-linear transformation of the input space $\mathcal{X}$. The learned function will be non-linear in $\mathcal{X}$, and may have enhanced representational power. Instead of explicitly transforming the data before model learning, the so-called "kernel trick" provides a computational shortcut, which makes it possible to learn the same model using only operations on the data in $\mathcal{X}$. This is achieved by defining

an appropriate function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, known as a kernel. It is required for most learning tasks that k be a *Mercer kernel*:[1]

Definition 5.2.1. *Mercer kernel*
A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a Mercer kernel if the following properties hold:

- $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$ *for all* $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ *(symmetric);*
- $\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ *for all* $n \in \mathbb{N}^+$, $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ *and* $\alpha_i, \alpha_j \in \mathbb{R}$ *(positive semi-definite).*

By Mercer's Theorem (Mercer, 1909), such a function implicitly defines a (not necessarily unique) mapping from the input space $\mathcal{X}$ into a Hilbert space $\mathcal{H}$; that is, there exists $\phi : \mathcal{X} \to \mathcal{H}$, such that

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}},$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$ denotes the inner-product in $\mathcal{H}$. Further, the Moore–Aronszajn Theorem (Aronszajn, 1950) states that the mapping $\boldsymbol{x} \mapsto k(\boldsymbol{x}, \cdot)$ uniquely defines a reproducing kernel Hilbert space $\mathcal{H}_k$ of functions.

Definition 5.2.2. *RKHS*
A reproducing kernel Hilbert space is a Hilbert Space $\mathcal{H}_k$ that possesses a reproducing kernel, i.e. a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for which the following hold:

- $k(\boldsymbol{x}, \cdot) \in \mathcal{H}_k$ *for all* $\boldsymbol{x} \in \mathcal{X}$*;*
- $\langle f, k(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}_k} = f(\boldsymbol{x})$ *for all* $\boldsymbol{x} \in \mathcal{X}$, $f \in \mathcal{H}_k$ *(reproducing property).*

Hence, we do not need to start with a specific feature transformation in mind. For symbolic objects such as graphs or strings, it is not obvious how to construct a transformation into a high-dimensional vector space. It is often easier to construct a symmetric, positive-semidefinite function between objects, which implicitly defines a (possibly infinite-dimensional) vector representation.

[1] For a discussion on learning with non-Mercer kernels, see Ong et al. (2004).

5.2.2 Vector-valued RKHSs

Vector-valued RKHSs were introduced to the machine learning community by Micchelli and Pontil (2005) as a way to extend kernel machines from scalar to vector outputs, with the goal of learning nonlinear vector-valued functions. As with scalar-valued kernel machines, the idea is to transform the input space $\mathcal{X}$ into a higher-dimensional space, wherein a linear vector-valued function can be learned. In this case, we require an operator-valued kernel[2] function K:

Definition 5.2.3. *Operator-valued kernel*
An $\mathcal{L}(\mathcal{Z})$-valued kernel K on $\mathcal{X}^2$ is a function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Z})$;

- *K is Hermitian if $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}', \boldsymbol{x})^*$ for any $(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{X}^2$, where * denotes the adjoint operator;*
- *K is positive-semidefinite on $\mathcal{X}$ if K is Hermitian and for every $n \in \mathbb{N}^+$ and $\{(\boldsymbol{x}_i, \boldsymbol{z}_i) \in \mathcal{X} \times \mathcal{Z}\}_{i=1}^n$,*

$$\sum_{i,j} \langle \boldsymbol{z}_i, K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{z}_j \rangle_{\mathcal{Z}} \geq 0.$$

Using an $\mathcal{L}(\mathcal{Z})$-valued positive-semidefinite kernel, we can construct the set of functions $\{K(\cdot, \boldsymbol{x})\boldsymbol{z} : \boldsymbol{x} \in \mathcal{X}, \boldsymbol{z} \in \mathcal{Z}\}: \mathcal{X} \to \mathcal{Z}$. Denote by $\mathcal{H}_0$ the unique linear space formed by the span of these functions. We make $\mathcal{H}_0$ a pre-Hilbert space by defining the inner product

$$\langle f, f' \rangle_{\mathcal{H}_0} = \sum_{i,j=1}^{N} \langle \boldsymbol{z}_i, K(\boldsymbol{x}_i, \boldsymbol{x}'_j)\boldsymbol{z}'_j \rangle_{\mathcal{Z}},$$

for $f = \sum_{i=1}^N K(\boldsymbol{x}_i, \cdot)\boldsymbol{z}_i$, $f' = \sum_{i=1}^N K(\boldsymbol{x}'_i, \cdot)\boldsymbol{z}'_i \in \mathcal{H}_0$. This can always be completed into the ($\mathcal{Z}$-valued reproducing kernel) Hilbert space $\mathcal{H}_K$ (Micchelli and Pontil, 2005; Minh et al., 2016).

[2]We capitalise the operator-valued kernel function $K(\cdot, \cdot)$ to distinguish it from scalar-valued kernels $k(\cdot, \cdot)$. For both type of kernel, the corresponding Gram matrix is denoted using capitalised boldface (e.g. $\boldsymbol{K}$)—the type of kernel function underlying the matrix should be clear from context.

Definition 5.2.4. *Vector-valued RKHS*
A vector-valued reproducing kernel Hilbert space is a Hilbert Space $\mathcal{H}_K$ of functions $f : \mathcal{X} \to \mathcal{Z}$ that possesses an operator-valued reproducing kernel, i.e. a positive-semidefinite function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Z})$ for which the following hold:

- *$K(\boldsymbol{x}, \cdot)\boldsymbol{z} \in \mathcal{H}_K$ for all $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{z} \in \mathcal{Z}$;*
- *$\langle f, K(\boldsymbol{x}, \cdot)\boldsymbol{z} \rangle_{\mathcal{H}_K} = \langle f(\boldsymbol{x}), \boldsymbol{z} \rangle_{\mathcal{Z}}$ for all $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{z} \in \mathcal{Z}$ and $f \in \mathcal{H}_K$ (reproducing property).*

Note that from the reproducing property, we can consider a feature mapping $\Phi : \mathcal{X} \times \mathcal{Z} \to \mathcal{H}_K$ defined by $\Phi(\boldsymbol{x}, \boldsymbol{z}) \coloneqq K(\cdot, \boldsymbol{x})\boldsymbol{z}$. Then:

$$\begin{aligned} \langle \boldsymbol{z}, K(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{z}' \rangle_{\mathcal{Z}} &= \langle K(\cdot, \boldsymbol{x})\boldsymbol{z}, K(\cdot, \boldsymbol{x}')\boldsymbol{z}' \rangle_{\mathcal{H}_K} \\ &= \langle \Phi(\boldsymbol{x}, \boldsymbol{z}), \Phi(\boldsymbol{x}', \boldsymbol{z}') \rangle_{\mathcal{H}_K}. \end{aligned}$$

Hence, as in the scalar-valued case, the kernel can be used to compute an inner-product in some (possibly infinite-dimensional) feature space. The usual kernel trick can be recovered when $\mathcal{Z} = \mathbb{R}$.

5.2.3 Multi-view Kernels

In what follows, our goal will be to learn a $\mathcal{Z}$-valued function using multi-view data. We assume there are v different views of the set of n observations; that is, $\mathcal{Z} = \mathbb{R}^v$ and $\mathcal{X} = \mathcal{X}^{(1)} \times \ldots \times \mathcal{X}^{(v)}$. The learning algorithm will be based on computing and manipulating the kernel Gram matrix associated with an operator-valued kernel function. It is not immediately obvious how to define a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{v \times v}$, which takes two objects—each having multiple feature representations—and outputs a matrix. In Section 5.3.3 and Section 6.3, we will discuss how to construct such a function using individual scalar-valued kernel functions, which can be chosen appropriate to the data type of each view. Crucially, the resulting function must be positive-semidefinite in order to validate the following theorem, the proof of which can be found in Kadri et al. (2016).

Theorem 1. *An $\mathcal{L}(\mathcal{Z})$-valued kernel K on $\mathcal{X} \times \mathcal{X}$ is the reproducing kernel of some Hilbert space $\mathcal{H}_K$, if and only if K is positive-semidefinite.*

This bijection allows us to build a general learning scheme that can incorporate information both within and between data of disparate representation. We now describe that learning framework.

5.3 Multi-view Semi-supervised Structural Risk Minimisation

In this section, we present the regularisation framework for multi-view semi-supervised learning. We begin by describing a multi-view extension of the classical structural risk minimisation (SRM) framework (Vapnik and Chervonenkis, 1974), which is foundational to many well-known supervised learning schemes. This forms the basis for both the semi-supervised classifier described in this chapter and the general multi-view supervised learning scheme of Chapter 6. First we describe the general setting for an arbitrary convex loss function. We then construct a probabilistic classifier using the negative log-likelihood of the Bernoulli distribution and derive the algorithm's optimisation procedure.

5.3.1 Supervised Learning Framework

The v-view supervised learning problem involves using a labelled training set $\{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^n$ to estimate a function $g\colon \mathcal{X} \to \mathcal{Y}$, where—unlike the single-view setting—the input space is given by $\mathcal{X} = \mathcal{X}^{(1)} \times \ldots \times \mathcal{X}^{(v)}$, with $\mathcal{X}^{(s)}, \mathcal{X}^{(t)}$ possibly disjoint spaces for $s \neq t$. In order to deal with multiple feature representations, we define the target function as $g(\cdot) \coloneqq w(f(\cdot))$, where a bounded linear operator $w : \mathbb{R}^v \to \mathbb{R}$ is used to transform the output of a vector-valued function $f\colon \mathcal{X} \to \mathbb{R}^v$ into a prediction. Here we take the arithmetic average of the output[3] by choosing $g(\cdot) = \boldsymbol{w}^\top(\boldsymbol{b} + f(\cdot))$, for a weight vector $\boldsymbol{w} = \left(\frac{1}{v}, \ldots, \frac{1}{v}\right) \in \mathbb{R}^v$ and a vector of intercept, or bias, terms $\boldsymbol{b} \in \mathbb{R}^v$.

[3]Note, other weights or combination schemes are possible—for the generalised learning framework we develop in Chapter 6, we allow these weights to be learned.

In what follows, we appeal to the SRM approach to supervised learning, trading off the complexity of the learned function against its success at fitting the training data. In this way, the chosen function does not overfit and can generalise to new data. We can express this as minimisation of the following objective function:

$$J(\boldsymbol{b}, f) = \sum_{i=1}^{n} V\left(y_i, \boldsymbol{w}^\top(\boldsymbol{b} + f(\boldsymbol{x}_i))\right) + \lambda \|f\|^2_{\mathcal{H}_K}, \tag{5.1}$$

where $V\colon \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$ is some convex loss function, and a scalar λ controls how strongly to penalise complexity of f through the squared L^2-norm $\|f\|^2_{\mathcal{H}_K}$.

In (5.1), the hypothesis space of functions f is chosen to be the vector-valued RKHS $\mathcal{H}_K$ associated with an operator-valued kernel K. As $\mathcal{H}_K$ is possibly infinite-dimensional, it is not obvious that this optimisation problem can be solved efficiently. The following Representer Theorem (Micchelli and Pontil, 2005) proves that the minimiser lies in the subspace of $\mathcal{H}_K$ spanned by the images of the training data.

Theorem 2. *Let K be a positive-semidefinite operator-valued kernel and $\mathcal{H}_K$ its associated vector-valued RKHS. The solution $\widehat{f} \in \mathcal{H}_K$ of the regularised optimisation problem* (5.1) *has the following form:*

$$\widehat{f} = \arg\min_f J(f \ ; \ \boldsymbol{b}) = \sum_{j=1}^{n} K(\cdot, \boldsymbol{x}_j)\boldsymbol{c}_j, \ \textit{with } \boldsymbol{c}_j \in \mathbb{R}^v. \tag{5.2}$$

We can re-write the norm $\|f\|^2_{\mathcal{H}_K}$ of a function of this form as:

$$\begin{aligned}
\|f\|^2_{\mathcal{H}_K} &= \langle f, f \rangle_{\mathcal{H}_K} \\
&= \left\langle \sum_{i=1}^{n} K(\cdot, \boldsymbol{x}_i)\boldsymbol{c}_i, \sum_{j=1}^{n} K(\cdot, \boldsymbol{x}_j)\boldsymbol{c}_j \right\rangle_{\mathcal{H}_K} \\
&= \sum_{i,j=1}^{n} \langle \boldsymbol{c}_i, K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{c}_j \rangle_{\mathbb{R}^v} ,
\end{aligned}$$

where the last equality is a result of the reproducing property of K. We can therefore simplify the infinite-dimensional optimisation problem (5.1) to a

finite dimensional one:

$$J(\boldsymbol{b}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) = \sum_{i=1}^{n} V\left(y_i, \boldsymbol{w}^\top \left[\boldsymbol{b} + \sum_{j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{c}_j\right]\right) + \lambda \sum_{i,j=1}^{n} \langle \boldsymbol{c}_i, K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{c}_j \rangle .$$

Let $\boldsymbol{K} \in \mathbb{R}^{nv \times nv}$ denote the Gram matrix of $K(\cdot, \cdot)$ and $\boldsymbol{c} \in \mathbb{R}^{nv}$ be the vector formed by concatenating the elements of $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ such that $\boldsymbol{c}$ is formed of v vectors of length n. The problem can then be written more concisely as:

$$J(\boldsymbol{b}, \boldsymbol{c}) = V\left(\boldsymbol{y}, (\boldsymbol{w}^\top \otimes \boldsymbol{I}_n)(\boldsymbol{b} \otimes \boldsymbol{1}_n + \boldsymbol{K}\boldsymbol{c})\right) + \lambda \boldsymbol{c}^\top \boldsymbol{K}\boldsymbol{c}, \tag{5.3}$$

where $\otimes$ denotes the Kronecker product. For a given loss function V, this problem can typically be solved using gradient descent methods.

5.3.2 Relation to Single-view Learning Schemes

We can recover several well-known learning schemes as special cases when $v = 1$. For example, choosing a squared error loss function leads to the Regularised Least Squares algorithm (Rifkin et al., 2003), while the Support Vector Machine (SVM) (Cortes and Vapnik, 1995)—though traditionally framed geometrically as a method for finding a hyperplane that achieves the largest separation between two classes—corresponds exactly to the single-view version of (5.3) with the hinge loss function (Evgeniou et al., 2000).

5.3.3 Semi-supervised Learning

As discussed in Section 5.1, the field of multi-view learning has its roots in semi-supervised learning, where the aim is to estimate some unknown target function by utilising a collection of unlabelled data in conjunction with the labelled training dataset.

Say we have m labelled observations $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ and the remainder are unlabelled $\{\boldsymbol{x}_i\}_{i=m+1}^{n}$, where often $m \ll n$. So-called *co-regularisation* algorithms

begin with a kernel machine that makes separate view-specific predictions on *all* available data; an additional regularisation term is then added that explicitly penalises disagreement between the predictions made by the individual functions. Early works, which were limited to two views (Sindhwani and Rosenberg, 2008; Sindhwani et al., 2005), used the sum of squared errors between predictions as the regularisation term. We can achieve the same for an arbitrary number of views by modifying the objective function (5.1) to:

$$\begin{aligned} J(\boldsymbol{b}, f) = &\sum_{i=1}^{m} V\left(y_i, \boldsymbol{w}^\top(\boldsymbol{b} + f(\boldsymbol{x}_i))\right) + \lambda \|f\|^2_{\mathcal{H}_K} \\ &+ \phi \sum_{i=1}^{n} \sum_{s,t=1;\ s<t}^{v} \left\| f^{(s)}(\boldsymbol{x}_i) - f^{(t)}(\boldsymbol{x}_i) \right\|^2, \end{aligned} \tag{5.4}$$

where we note that the loss function is only applied over the m labelled samples and the added regularisation term (controlled by scalar ϕ) is applied to all n data. Minh et al. (2016) prove a Representer theorem that shows that the minimiser is a combination of kernel products evaluated on both the labelled and unlabelled data:

$$\widehat{f} = \sum_{j=1}^{n} K(\cdot, \boldsymbol{x}_j)\boldsymbol{c}_j.$$

We can construct an operator-valued kernel function K such that the v-dimensional prediction

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{c}_j$$

represents a prediction made using each of the v views. Given scalar-valued kernel functions $k_s(\cdot, \cdot)$, $s = 1, \ldots, v$, which can be suitably chosen to represent some notion of similarity between observations in each view, let

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j)_{lm} = \begin{cases} k_l(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ if } l = m, \\ 0 \text{ otherwise.} \end{cases}$$

The Gram matrix $\boldsymbol{K}$ is block-diagonal and each block corresponds to the scalar-valued kernel Gram matrix over one view.

Thus, let $\boldsymbol{f} = \boldsymbol{K}\boldsymbol{c} \in \mathbb{R}^{nv}$ denote the vector of (vector-valued) predictions of all training samples. As in Minh et al. (2016), we can construct an operator $\boldsymbol{M}_B$ such that the between-view regularisation term in (5.4) can be expressed by the semi-norm $\langle \boldsymbol{f}, \boldsymbol{M}_B \boldsymbol{f} \rangle$. In their formulation, $\boldsymbol{f}$ is arranged differently to how we have defined it. However, we can define an analogous operator, which is suitable for our framework. Let

$$\boldsymbol{M}_v = v\boldsymbol{I}_v - \mathbf{1}_v \mathbf{1}_v^\top.$$

This is the $v \times v$ matrix with $(v-1)$ on the diagonal and -1 elsewhere. Then for a given training prediction $f(\boldsymbol{x}_i) = (f^{(1)}(\boldsymbol{x}_i), \ldots, f^{(v)}(\boldsymbol{x}_i))^\top \in \mathbb{R}^v$,

$$f(\boldsymbol{x}_i)^\top \boldsymbol{M}_v f(\boldsymbol{x}_i) = \sum_{s,t=1,\ s<t}^{v} \left\| f^{(s)}(\boldsymbol{x}_i) - f^{(t)}(\boldsymbol{x}_i) \right\|^2.$$

For the regularisation term, we require an operator that can sum the above term over all training data points. Denote by $\boldsymbol{M}_{v,c}$ the cth column of $\boldsymbol{M}_v$. We can construct $\boldsymbol{M}_B$ as the following block matrix:

$$\boldsymbol{M}_B = \begin{bmatrix} \boldsymbol{I}_n \otimes \boldsymbol{M}_{v,1}^\top \\ \boldsymbol{I}_n \otimes \boldsymbol{M}_{v,2}^\top \\ \vdots \\ \boldsymbol{I}_n \otimes \boldsymbol{M}_{v,v}^\top \end{bmatrix}. \tag{5.5}$$

We then have that

$$\boldsymbol{f}^\top \boldsymbol{M}_B \boldsymbol{f} = \sum_{i=1}^{n} \sum_{s,t=1;\ s<t}^{v} \left\| f^{(s)}(\boldsymbol{x}_i) - f^{(t)}(\boldsymbol{x}_i) \right\|^2,$$

as required. We can express the objective function as:

$$\begin{aligned} J(\boldsymbol{b}, \boldsymbol{c}) = V\left(\boldsymbol{y}, (\boldsymbol{w}^\top \otimes \boldsymbol{I}_m)(\boldsymbol{b} \otimes \mathbf{1}_m + \boldsymbol{K}_m \boldsymbol{c})\right) + \lambda \boldsymbol{c}^\top \boldsymbol{K} \boldsymbol{c} \\ + \phi\langle \boldsymbol{K}\boldsymbol{c}, \boldsymbol{M}_B \boldsymbol{K}\boldsymbol{c} \rangle, \end{aligned} \tag{5.6}$$

where $\boldsymbol{K}_m \in \mathbb{R}^{mv \times nv}$ is the submatrix of $\boldsymbol{K} \in \mathbb{R}^{nv \times nv}$ consisting of the rows that correspond to the labelled observations.

5.4 Semi-supervised Kernel Logistic Regression

In this section we use the framework just described to construct a multi-view binary classifier that outputs probabilistic predictions. This is achieved by replacing the loss function in Equation (5.6) with the negative log-likelihood of the Bernoulli distribution (Zhu and Hastie, 2001):

$$\begin{aligned} J(\boldsymbol{b}, \boldsymbol{c}) = & -\sum_{i=1}^{m} \left(y_i g(\boldsymbol{x}_i) - \log(1 + \exp(g(\boldsymbol{x}_i)))\right) + \lambda \boldsymbol{c}^\top \boldsymbol{K} \boldsymbol{c} \\ & + \phi \langle \boldsymbol{K}\boldsymbol{c}, \boldsymbol{M}_B \boldsymbol{K} \boldsymbol{c} \rangle, \end{aligned}$$

where, as before, $g(\boldsymbol{x}_i) = \boldsymbol{w}^\top(\boldsymbol{b} + f(\boldsymbol{x}_i))$. Denote the vector of predictions for all m labelled training samples as

$$\boldsymbol{g} = (\boldsymbol{w}^\top \otimes \boldsymbol{I}_m)(\boldsymbol{b} \otimes \boldsymbol{1}_m + \boldsymbol{K}_m \boldsymbol{c}) \in \mathbb{R}^m. \tag{5.7}$$

This allows us to write

$$J(\boldsymbol{b}, \boldsymbol{c}) = -\boldsymbol{y}^\top \boldsymbol{g} + \boldsymbol{1}_m^\top \log(\boldsymbol{1}_m + \exp(\boldsymbol{g})) + \lambda \boldsymbol{c}^\top \boldsymbol{K} \boldsymbol{c} + \phi \langle \boldsymbol{K}\boldsymbol{c}, \boldsymbol{M}_B \boldsymbol{K} \boldsymbol{c} \rangle. \tag{5.8}$$

The dual model parameters $\boldsymbol{\alpha} \coloneqq (\boldsymbol{b}, \boldsymbol{c}) \in \mathbb{R}^{(n+1)v}$ can be estimated using an iteratively re-weighted least squares procedure. Let $\boldsymbol{V} \coloneqq (\boldsymbol{1}_m \otimes \boldsymbol{I}_v, \boldsymbol{K}_m) \in \mathbb{R}^{mv \times (n+1)v}$ and define $\boldsymbol{U},\ \boldsymbol{P} \in \mathbb{R}^{(n+1)v \times (n+1)v}$ as follows:

$$\boldsymbol{U} \coloneqq \begin{bmatrix} \boldsymbol{0}_{v \times v} & \boldsymbol{0}_{v \times nv} \\ \boldsymbol{0}_{nv \times v} & \boldsymbol{K} \end{bmatrix}, \qquad \boldsymbol{P} \coloneqq \begin{bmatrix} \boldsymbol{0}_{v \times v} & \boldsymbol{0}_{v \times nv} \\ \boldsymbol{0}_{nv \times v} & \boldsymbol{K}^\top \boldsymbol{M}_B^\top \boldsymbol{K} \end{bmatrix},$$

i.e. the matrices $\boldsymbol{K}$ and $\boldsymbol{K}^\top \boldsymbol{M}_B^\top \boldsymbol{K}$ padded above and to the left with v rows/columns of zeros. To simplify notation in what follows, further let $\boldsymbol{A} \coloneqq (\boldsymbol{w}^\top \otimes \boldsymbol{I}_m)\boldsymbol{V} \in \mathbb{R}^{m \times (n+1)v}$. Then (5.7) and (5.8) can be expressed in

terms of $\boldsymbol{\alpha}$:

$$\boldsymbol{g} = \boldsymbol{A}\boldsymbol{\alpha};$$

$$J(\boldsymbol{\alpha}) = -\boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{\alpha} + \mathbf{1}_m^\top \log(\mathbf{1}_m + \exp(\boldsymbol{A}\boldsymbol{\alpha})) + \lambda \boldsymbol{\alpha}^\top \boldsymbol{U}\boldsymbol{\alpha} + \phi \boldsymbol{\alpha}^\top \boldsymbol{P}\boldsymbol{\alpha}. \quad (5.9)$$

The gradient is thus:

$$\nabla_{\boldsymbol{\alpha}} = -\boldsymbol{A}^\top(\boldsymbol{y} - \boldsymbol{\mu}) + 2\lambda \boldsymbol{U}\boldsymbol{\alpha} + \phi \boldsymbol{P}\boldsymbol{\alpha} + \phi \boldsymbol{P}^\top \boldsymbol{\alpha}, \quad (5.10)$$

where

$$\boldsymbol{\mu} = \frac{\exp(\boldsymbol{A}\boldsymbol{\alpha})}{\mathbf{1}_m + \exp(\boldsymbol{A}\boldsymbol{\alpha})}.$$

The Hessian matrix is given by:

$$\nabla^2_{\boldsymbol{\alpha}} = \boldsymbol{A}^\top \boldsymbol{W} \boldsymbol{A} + 2\lambda \boldsymbol{U} + \phi \boldsymbol{P} + \phi \boldsymbol{P}^\top, \quad (5.11)$$

where $\boldsymbol{W} = \text{diag}(\mu_1(1-\mu_1), \ldots, \mu_m(1-\mu_m))$. The Newton-Raphson method then updates the model parameters according to:

$$\widehat{\boldsymbol{\alpha}}_{t+1} = \widehat{\boldsymbol{\alpha}}_t - \left(\nabla^2_{\boldsymbol{\alpha}}\right)^{-1} \nabla_{\boldsymbol{\alpha}}. \quad (5.12)$$

Let $\boldsymbol{Q} \coloneqq 2\lambda \boldsymbol{U} + \phi \boldsymbol{P} + \phi \boldsymbol{P}^\top$. Then substituting (5.10) and (5.11) into (5.12) gives:

$$\widehat{\boldsymbol{\alpha}}_{t+1} = \widehat{\boldsymbol{\alpha}}_t + \left[\boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{A} + \boldsymbol{Q}\right]^{-1} \left[\boldsymbol{A}^\top [\boldsymbol{y} - \widehat{\boldsymbol{\mu}}_t] - \boldsymbol{Q}\widehat{\boldsymbol{\alpha}}_t\right],$$

which can be rearranged to form

$$\widehat{\boldsymbol{\alpha}}_{t+1} = \left[\boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{A} + \boldsymbol{Q}\right]^{-1} \boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{z}_t, \quad (5.13)$$

where $\boldsymbol{z}_t = \boldsymbol{A}\widehat{\boldsymbol{\alpha}}_t + \boldsymbol{W}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$.

The algorithm proceeds by iteratively updating $\widehat{\boldsymbol{\alpha}}_t$ until some convergence criterion is met. A simple stopping rule is to compare the value of the objective function in consecutive iterations and deem the algorithm converged when the ratio

$$\frac{|J(\widehat{\boldsymbol{\alpha}}_t) - J(\widehat{\boldsymbol{\alpha}}_{t-1})|}{|J(\widehat{\boldsymbol{\alpha}}_t)|} < \epsilon,$$

for some small scalar $\epsilon > 0$. Pseudocode for the algorithm is presented in

Algorithm 1.

5.5 Experiments

In this section, we present experimental results using the multi-view semi-supervised learning scheme just presented. First, using synthetically generated data, we demonstrate the algorithm's ability to improve accuracy by harnessing unlabelled observations. We then test the model on a real-world malware classification dataset.

5.5.1 Simulated Data

In this experiment, we simulate a toy two-view, two-class dataset. This is used to investigate the efficacy of our semi-supervised learning algorithm by performing classification both with and without the inclusion of unlabelled observations.

Data Generation

In order to aid visualisation, we base our multi-view data on a 'two circle' dataset, which is commonly used for demonstrating non-linear regression or clustering algorithms. For the first view, we simulate 1,000 points from two circles in $\mathbb{R}^2$ (representing two classes) and we add Gaussian noise. Each point represents an observation from one of two classes, corresponding to the circle from which the point was drawn. A plot of the data is given in Figure 5.1a. This type of data is often found in introductory texts on kernel methods; the two classes are not linearly separable in their original feature space, though they easily become linearly separable when transformed into a higher-dimensional space (e.g. in $\mathbb{R}^3$ via a polynomial kernel).

We generate the second view of data by applying the following non-linear

Algorithm 1: Multi-view Semi-supervised Kernel Logistic Regression

Input:

$\boldsymbol{y} \in \{0, 1, \texttt{NA}\}^n$: vector of responses (`NA` indicates unlabelled)

$\boldsymbol{K} \in \mathbb{R}^{nv \times nv}$: $\mathcal{L}(\mathbb{R}^v)$-valued kernel Gram matrix

$\lambda, \phi \in \mathbb{R}$: regularisation parameters

`max_iter` $\in \mathbb{N}$: (optional) maximum number of iterations

$\epsilon \in \mathbb{R}$: convergence threshold

Initialise:

$\widehat{\boldsymbol{\alpha}}_0 \leftarrow \mathbf{0}_{(n+1)v}$

Procedure:

Construct $\boldsymbol{K}_m \in \mathbb{R}^{mv \times nv}$ from m labelled rows of $\boldsymbol{K}$

Construct $\boldsymbol{M}_B$ from (5.5)

Construct $\boldsymbol{V}$, $\boldsymbol{U}$ and $\boldsymbol{P}$ as in Section 6.2.3

$\boldsymbol{A} = (\boldsymbol{w}^\top \otimes \boldsymbol{I}_m)\boldsymbol{V}$

$\boldsymbol{Q} = 2\lambda \boldsymbol{U} + \phi \boldsymbol{P} + \phi \boldsymbol{P}^\top$

$\boldsymbol{w} = \frac{1}{v}\mathbf{1}_v$

for $t = 1$ *to* `max_iter` **do**

$\boldsymbol{\mu}_t = \exp(\boldsymbol{A}\widehat{\boldsymbol{\alpha}}_{t-1}) / (\mathbf{1} + \exp(\boldsymbol{A}\widehat{\boldsymbol{\alpha}}_{t-1}))$

$\boldsymbol{W}_t = \operatorname{diag}(\mu_{t,1}(1-\mu_{t,1}), \ldots, \mu_{t,m}(1-\mu_{t,m}))$

$\boldsymbol{z}_t = \boldsymbol{A}\widehat{\boldsymbol{\alpha}}_{t-1} + \boldsymbol{W}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$

$\widehat{\boldsymbol{\alpha}}_t \leftarrow \left[\boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{A} + \boldsymbol{Q}\right]^{-1} \boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{z}_t$

`# check convergence`

$J(\widehat{\boldsymbol{\alpha}}_t) = -\boldsymbol{y}^\top \boldsymbol{A}\widehat{\boldsymbol{\alpha}}_t + \mathbf{1}^\top \log(\mathbf{1} + \exp(\boldsymbol{A}\widehat{\boldsymbol{\alpha}}_t)) + \lambda \widehat{\boldsymbol{\alpha}}_t^\top \boldsymbol{U}\widehat{\boldsymbol{\alpha}}_t + \phi \widehat{\boldsymbol{\alpha}}_t^\top \boldsymbol{P}\widehat{\boldsymbol{\alpha}}_t$

if $|J(\widehat{\boldsymbol{\alpha}}_t) - J(\widehat{\boldsymbol{\alpha}}_{t-1})| / |J(\widehat{\boldsymbol{\alpha}}_t)| < \epsilon$ **then**

break

Output:

Estimated parameter $\widehat{\boldsymbol{\alpha}}$

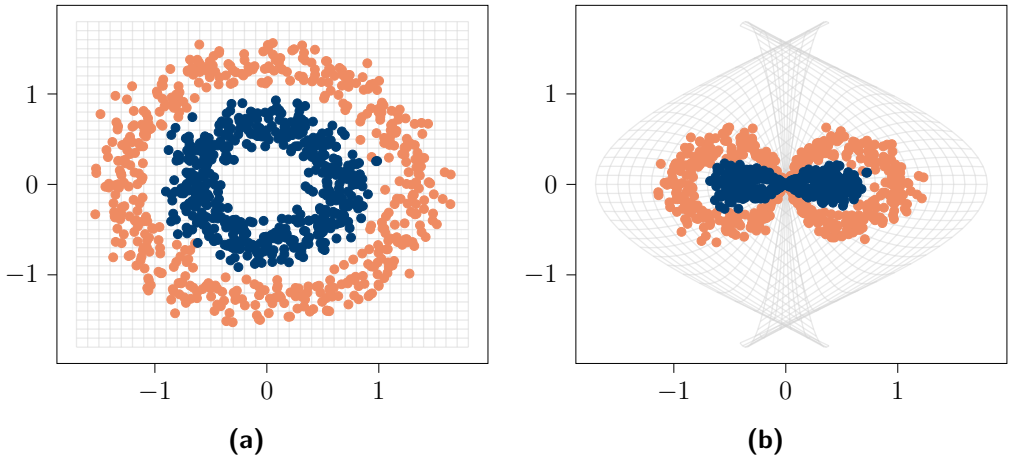


Figure 5.1: A synthetic multi-view dataset. View 1 (a) is the classic 'two circles' dataset. We generate view 2 (b) by applying a non-linear transformation (5.14) to view 1. To illustrate the effect of the transformation, we show a grid corresponding to view 1's coordinate system under the transformation.

transformation to each point in view 1:

$$t\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x\cos(y) \\ y\sin(x) \end{bmatrix}. \tag{5.14}$$

By doing so, we simulate a new view of the data, where the shape and separability of the two class clusters is different but we maintain a correspondence between points in the two spaces. This view is shown in Figure 5.1b.

Evaluation

To fit our model, we choose a scalar-valued kernel function for each view. For view 1, we use the following polynomial kernel with degree 2:

$$k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\boldsymbol{x}_i^{(1)\top}\boldsymbol{x}_j^{(1)} + 1\right)^2.$$

For view 2, we use the Gaussian, or radial basis function kernel:

$$k_2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i^{(2)} - \boldsymbol{x}_j^{(2)}\|^2}{2\sigma^2}\right), \tag{5.15}$$

with σ set to be 1.

In this toy dataset, neither of the two views of data should pose much of a challenge to a kernel machine. However, we are interested in whether our *multi-view* kernel machine can capture a dependence between the two views by favouring models that make the same predictions on *unlabelled* data. We randomly sample a tiny training set of only 5 observations and train the algorithm both with and without the inclusion of an additional 400 unlabelled observations. In both cases, the models were evaluated using a test set consisting of the remaining 595 data points. On this test set, the supervised and semi-supervised classifiers achieved an ROC AUC score of 75.3% and 95.8%, respectively.

To visually inspect the decision boundary in the original input space, we sample points along an evenly spaced grid in the coordinate space of view 1 and generate the corresponding points in view 2 via the non-linear transformation (5.14). We then make predictions for each multi-view point using the fitted supervised and semi-supervised learning models. A plot of the decision boundaries is shown in Figure 5.2. We can see clearly that in the semi-supervised case, despite having only a tiny number of labelled observations, the algorithm is able to learn a more accurate decision surface by capturing the dependence between the two views of data.

The synthetic dataset and experiment just presented are very artificial and designed to visually investigate and highlight the ability to learn from unlabelled data in the multi-view setting. As we see in the following section, the results are not necessarily an indication of how the algorithm can be expected to perform on more complex learning tasks in real-world applications.

5.5.2 Malware Detection

In this section, we apply our semi-supervised learning scheme to the problem of static malware detection. The term *static* refers to systems that attempt to classify programmes as malicious or benign without executing them, in contrast to *dynamic* malware detection techniques, which analyse a programme's

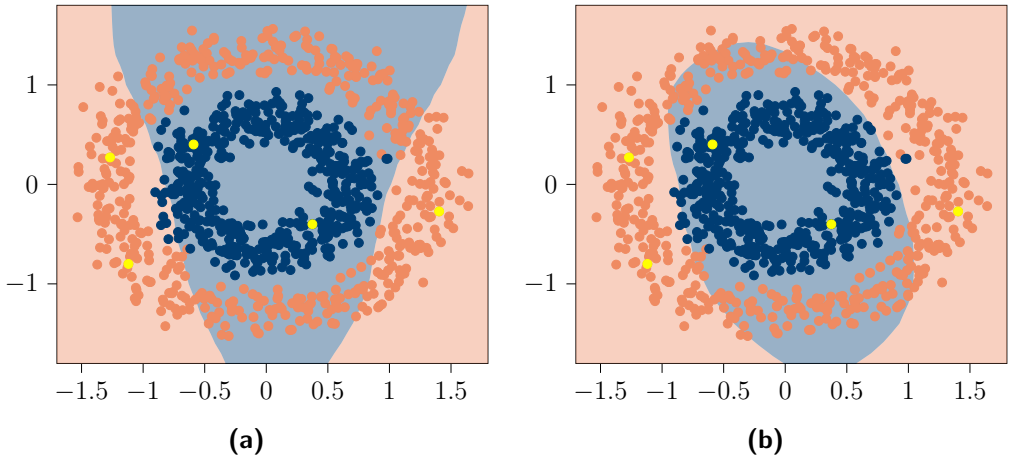


Figure 5.2: Decision boundaries of the fitted models: (a) supervised learning; (b) semi-supervised learning. The labelled observations used in the training of both models are shown in yellow.

runtime behaviour and time-dependent sequences of system calls (Gandotra et al., 2014; Ye et al., 2017). Although both types of approach have been proven to be undecidable in general (Cohen, 1987) (that is, it is impossible to construct an algorithm that always leads to a correct classification), static malware detection still offers an important protection layer in a security suite because, when successful, it allows malicious files to be detected prior to execution. A large body of machine learning research has focussed on static malware classification—see e.g. Kong and Yan (2013); Rieck et al. (2008); Schultz et al. (2000) or, for a review, Gandotra et al. (2014); Gibert et al. (2020).

Anderson et al. (2012) recognise malware detection as a multi-view learning problem; they argue that, while a malicious executable can disguise itself in some views, disguising itself in every view while maintaining malicious intent is substantially more difficult. It is also a promising application area for semi-supervised learning in particular, as there is a relative scarcity of labelled data and new unlabelled data becomes available continuously.

For our experiment, we use the EMBER dataset, described in Section 2.3.2. For each of the three modalities, we construct a scalar-valued kernel as follows:

- **Numeric features:** For each file, we construct a vector $\boldsymbol{x}_i^{(1)} \in \mathbb{R}^{12}$ using the features described in Table 2.3. All features are standardised to have zero mean and standard deviation equal to 1. We then use the radial basis function kernel:

$$k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i^{(1)} - \boldsymbol{x}_j^{(1)}\|^2}{2\sigma^2}\right),$$

  with σ set to be the mean distance between samples:

$$\sigma = \frac{1}{n(n-1)} \sum_{i}^{n} \sum_{j>i} \|\boldsymbol{x}_i^{(1)} - \boldsymbol{x}_j^{(1)}\|.$$

- **Histograms:** For our second view, we consider the byte histograms. As all the histograms contain the same number of bins and are normalised, each can be represented as a vector $\boldsymbol{x}_i^{(2)} \in \mathbb{R}^{256}$. We use the histogram intersection kernel:

$$k_2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{256} \min\left(\boldsymbol{x}_{i,k}^{(2)}, \boldsymbol{x}_{j,k}^{(2)}\right).$$

  Barla et al. (2003) show that this has the required mathematical properties to be used as a kernel function.

- **Strings:** The imported and exported functions for each file are represented as a variable-length list of strings, as described in Section 2.3.2. To construct a kernel function between observations, we first tokenise all strings and create a tf-idf (term frequency times inverse-document frequency) Bag-of-Words feature representation (see e.g. Schütze et al. (2008, p. 117)). This results in $\boldsymbol{x}_i^{(3)} \in \mathbb{R}^p$, where p is the total number

of unique 'words', i.e. imported/exported functions. We then use the cosine similarity kernel:

$$k_3(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^{(3)\top} \boldsymbol{x}_j^{(3)}}{\|\boldsymbol{x}_i^{(3)}\| \|\boldsymbol{x}_j^{(3)}\|}.$$

Evaluation

For our experiment, we selected a subset of 5,000 observations from the EMBER dataset. 500 observations were used as a training set, and the remainder held out for validation. To assess the efficacy of semi-supervised learning for malware detection, we repeated the following experiment, varying m, the number of labelled training data points:

1. A model is fit using only m labelled training points. We first perform hyperparameter tuning using a grid search over a range of values of λ. For each value, we carry out 5-fold cross validation. With each fold of the training set, we take a stratified sample of m points as labelled observations and discard the remaining points. The normalisation and tokenisation for the numeric and Bag-of-Words features are performed based on only these m observations. Once the optimal λ is found, this (supervised) model is evaluated on the validation set.

2. A model is fit using m labelled training points and $n = 500 - m$ unlabelled points. Again, 5-fold cross validation is used with a grid search to tune the hyperparameter ϕ. This time, having taken a stratified sample of m points as labelled observations, we remove the label from the remaining points. All observations are used for normalisation and tokenisation. The final (semi-supervised) model is evaluated on the validation set.

3. The ROC AUC score is used to compare classification results.

The results of the experiment are presented in Table 5.1. In Figure 5.3, we plot the ROC curves. Clearly, for very small values of m, the algorithm is

m	Supervised AUC	Semi-supervised AUC	Δ
5	76.87%	80.32%	+3.45%
10	81.79%	85.48%	+3.69%
20	83.35%	84.66%	+1.31%
40	88.46%	88.44%	−0.03%
80	91.89%	91.96%	+0.07%
250	95.31%	95.37%	+0.05%

Table 5.1: Results of the semi-supervised malware detection experiment. For varying values of m (the number of labelled observations) the ROC AUC scores for supervised and semi-supervised learning models are presented. Δ indicates the improvement achieved through the addition of unlabelled samples.

able to harness unlabelled data to improve predictive accuracy. However, the improvement in true positive rate, or *recall*, appears to only be achieved for rather high values of the false positive rate. Further, as we increase the number of labelled observations, the influence of the additional unlabelled data quickly diminishes.

5.6 Discussion

In this chapter, we explored the use of semi-supervised learning to incorporate information between multi-view data. Motivated by malware detection as a binary classification problem, we developed a logistic regression variant of the co-regularisation approach to multi-view learning (Minh et al., 2013). We proposed an efficient iteratively re-weighted least squares optimisation procedure and demonstrated, using both synthetic and real data, the algorithm's ability to simultaneously learn from multiple heterogeneous feature representations. Irrespective of the merits of utilising unlabelled data, the classification results are highly promising. Using only 250 training data points, the multi-view kernel machine is able to combine information from numeric features, histograms and strings to learn a model that achieves an AUC of over 95%. In the following chapter, we build on this framework to develop a novel general-purpose supervised learning scheme for multi-view data.

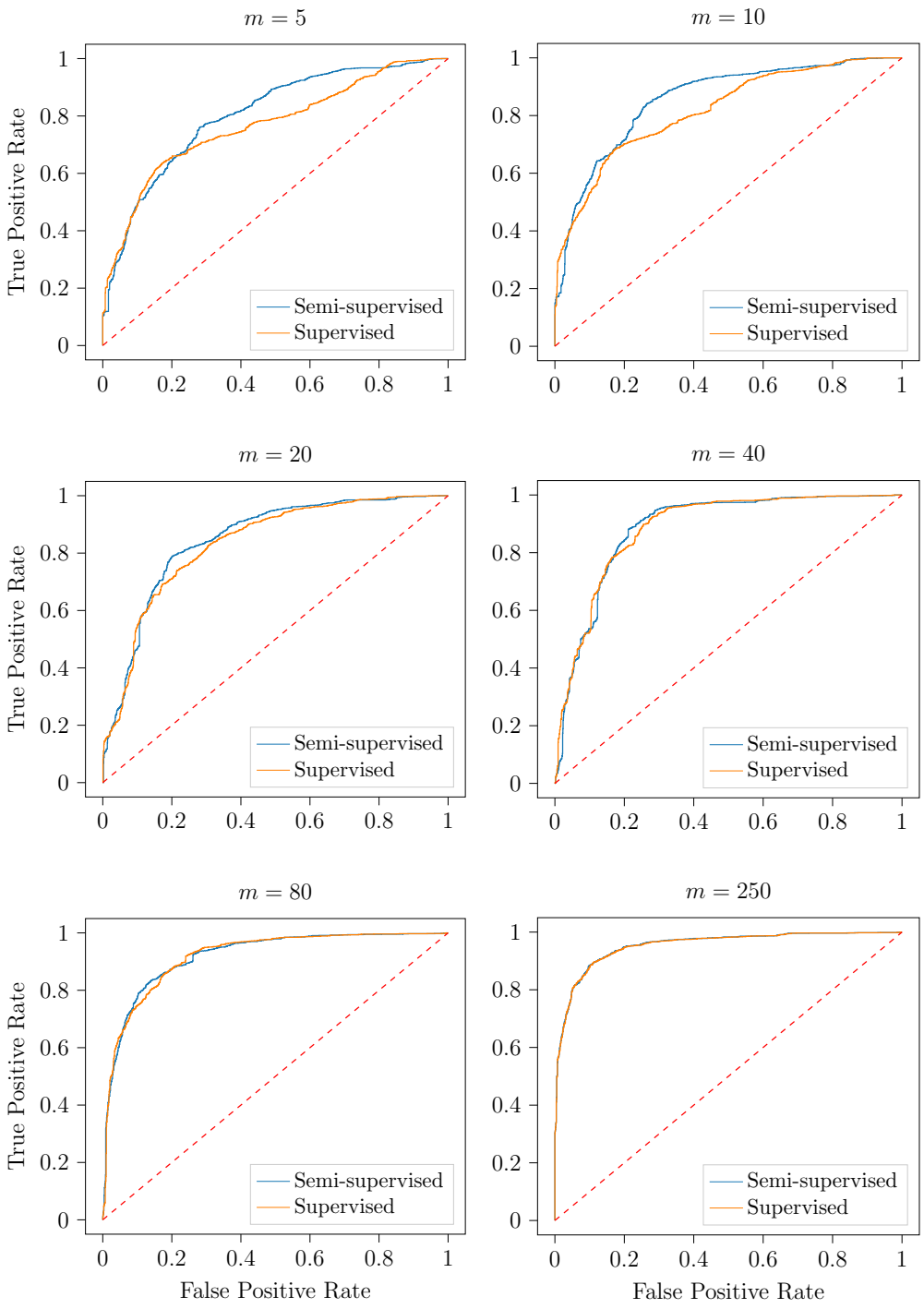


Figure 5.3: ROC curves for the malware detection experiment for varying values of m, the number of labelled observations.

6

Multi-view Generalised Kernel Machines

In Chapter 5, we presented a multi-view classifier designed to address some of the challenges of static malware detection: namely, the presence of heterogeneous feature modalities and a scarcity of labelled training data. Although our experimental results suggested that the benefits of incorporating unlabelled data may be limited, the framework itself showed great promise for its ability to be simultaneously tailored to multiple heterogeneous feature representations. This motivates us to extend the framework to build a general-purpose supervised learning scheme.

The Generalised Linear Model (GLM) (Nelder and Wedderburn, 1972) is one of the foundational supervised learning schemes and—due to its simplicity and flexibility—remains extremely widely used today. Implemented in every major statistical software platform, it allows a practitioner to fit a model to data with an arbitrary response type by simply selecting the appropriate link function from a wide family of candidates. We can imagine a similar framework for *multi-view* learning, in which fitting a model is as simple as specifying a suitable kernel for each modality and selecting the appropriate

link function for the response type. In Chapter 5, we were interested in a binary response type, so the algorithm we developed used the negative log-likelihood of the Bernoulli distribution as a loss function. Cawley et al. (2007) show that the negative log-likelihood of *any* exponential family distribution can be used as the loss function in a (SRM) kernel machine, thereby creating a non-linear variant of the GLM, which they term a *Generalised Kernel Machine* (GKM). In what follows, we adopt the same strategy to design an extremely general and highly flexible framework for multi-view learning that can be easily adapted to the distributional characteristics of a given response variable.

Our proposed formulation has many advantages over existing methods: it has a simple and general optimisation strategy for both classification and regression; an optimal (optionally sparse) weighting of views can be simultaneously learned; it produces interpretable predictions; it is flexible to a wide range of response types, including binary, categorical, discrete and continuous; and it provides outlier detection capabilities. We provide an iterative algorithm for implementing our learning scheme and demonstrate its performance and flexibility on real data. The content of this chapter is adapted from Hogan and Adams (under review 2023b).

The remainder of the chapter proceeds as follows: We begin in Section 6.1 by describing existing methods in the literature that are closely related to our proposed framework and discussing their shortcomings. To address these, in Section 6.2 we draw on GLMs to specify a unified learning scheme for both regression and classification. We describe in detail the optimisation procedure, an approach for detecting and penalising uninformative views, and the calculation of outlier scores. In Section 6.3, we describe several options for constructing multi-view kernel functions. In Section 6.4, we discuss different types of response variables that are commonly encountered and we present appropriate link functions for tailoring our proposed learning scheme to each. We perform real-data experiments in Section 6.5.

6.1 Background and Relevant Literature

The learning algorithm we develop in this chapter builds on the vector-valued kernel regularisation framework presented in Section 5.3. Due to its flexibility at modelling heterogeneous modalities through bespoke kernel functions, this has successfully been used as the basis for a number of multi-view learning methods. However, existing works—the most closely related of which being Huusari et al. (2019); Kadri et al. (2013); Minh et al. (2016)—suffer from a number of shortcomings.

First, these models have primarily focused on only two types of response data and two corresponding loss functions: typically, for regression, the squared error loss function; and for classification, the hinge loss function (resulting in an SVM). For many response data types (e.g. with restricted or skewed distributions), the quality and interpretability of predictions may be compromised. Despite this, in the more general machine learning literature, it is common to employ the squared error loss function regardless of whether the problem is one of regression or classification. Rifkin et al. (2003) point out that while squared error does not "make sense" for binary classification, empirical results on many datasets show that comparable predictive performance can be achieved, with the benefit of a simpler optimisation procedure. For statistical inference and outlier detection, such an argument is highly dissatisfactory. As in Section 5.4, we argue instead in favour of the logistic loss function, which leads to an interpretable model, the outputs of which are probabilistic predictions.

In many cases, the response variable is neither continuous nor binary. An example that is treated in the literature is multi-class data. Huusari et al. (2019) adopt a one-vs-all approach (see e.g. Webb (2003, p. 144)) to multi-class classification using squared error loss within each classifier. Such an approach suffers from multiple issues; for example, the scale of predictions made by each classifiers may differ, and each classifier may be trained on highly imbalanced datasets, even if the overall class distribution is balanced. Minh et al. (2016) present a generalisation of the simplex-cone SVM (Mroueh et al., 2012) for dealing with multi-class data. Simplex-cone SVMs have re-

ceived limited application, due to seemingly opaque interpretability (Pouliot, 2018). Again, we argue that this problem can be more satisfactorily solved by appealing to traditional statistical modelling approaches—in this case through multinomial logistic regression. By adopting a GLM approach, we generalise least squares, binary, multi-class and many other types of regression into a single unifying model with a straightforward optimisation scheme and interpretable outputs.

Another limitation of existing methods lies in how they combine the predictions made by different views. For example, Kadri et al. (2013) convert a scalar valued response variable to a vector with equal elements, and so all views are assumed to have equal importance, with no mechanism for estimating—or even assigning—weights to the contributions of various views. The inclusion of a combination operator by Minh et al. (2016) and Huusari et al. (2019) does allow for views to be weighted, though in all but the least-squares case, the authors fail to derive a method for learning these weights, deferring the challenge to future work. The learning scheme we propose in the following section is the only method capable of simultaneously learning the optimum weighting of views for a general response data type.

It is interesting to note that this sub-problem is the subject of an entire family of algorithms known as multiple kernel learning (MKL), which seek to find an optimal weighting of scalar-valued kernel functions and input this into a learning scheme such as an SVM. A number of these such algorithms were developed to address the need to consider multiple kernels, or parametrisations of kernels, to encode different notions of similarity among a single feature set (Crammer et al., 2003; Lanckriet et al., 2004a). MKL has also been used as a multi-view learning method to combine data from multiple sources (Bach et al., 2004; Lanckriet et al., 2004b; Stafford Noble, 2004). In Section 6.3, we describe how our proposed algorithm offers, as a special case, an elegant and straightforward solution to MKL.

6.2 A Unified Learning Scheme

In this section, we develop a learning scheme that is not only *capable* of handling different response types but *appropriate* for each. We adopt a GKM approach, such that the resulting framework encompasses standard regression and classification (both binary and multi-class), but is also suitable for other response types, e.g. count data, inter-arrival times, etc. This is particularly attractive for cyber security anomaly detection, where we do not have observations with binary labels $\mathcal{Y} \in \{\text{'benign'}, \text{'malicious'}\}$, but rather we wish to model some quantity of interest—often a count, e.g. number of bytes sent, number of distinct destination ports, etc.—and identify unusual observations. The model we develop has a general optimisation procedure. Simultaneous optimisation of the weight vector $\boldsymbol{w}$ (which in Chapter 5 was fixed) is trivial.

6.2.1 Generalised Linear Models

In a GLM, the vector of responses $\boldsymbol{y} = (y_1, \ldots, y_n)$ is assumed to be a realisation of a random variable Y, whose components are independently distributed with means $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$. A GLM consists of three components:

1. The *random component*: The components of Y are assumed to be independently distributed according to some exponential family distribution; that is, any distribution taking the form

$$f(y_i \; ; \; \theta_i, \phi) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{a_i(\phi)} + c(y_i, \phi)\right),$$

   for some functions $a_i(\cdot)$, $b(\cdot)$, $c(\cdot)$, where θ_i and ϕ are the parameters of the distribution.

2. The *systematic component*: In a standard GLM (i.e. single-view, "non-kernelised"), this takes the form $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}$ for some design matrix $\boldsymbol{X}$ and vector of estimated coefficients $\boldsymbol{\beta}$. In a GKM, we construct $\boldsymbol{\eta}$ in

the feature space $\mathcal{H}_K$:

$$\eta_i = \boldsymbol{w}^\top \left(\boldsymbol{b} + f(\boldsymbol{x}_i) \right).$$

3. The *link* function: A monotonic, differentiable function $h(\cdot)$ relates the systematic and random components, such that

$$\eta_i = h(\mu_i).$$

The link function $h(\cdot)$ is chosen to constrain the estimate of the conditional mean to lie within reasonable bounds for the particular member of the exponential family concerned. The link function such that $\theta_i = \eta_i$ is known as the canonical link and simplifies the optimisation procedure.

One appealing feature of the exponential family is that we can easily compute moments of the distribution by taking derivatives of the log-likelihood. The log-likelihood of an exponential family distribution can be written as:

$$\ell(\boldsymbol{\theta}, \phi \ ; \ \boldsymbol{y}) = \sum_{i=1}^{n} \frac{y_i \theta_i - b(\theta_i)}{a_i(\phi)} + c(y_i, \phi),$$

with partial derivatives

$$\frac{\partial \ell}{\partial \theta_i} = \frac{y_i - b'(\theta_i)}{a_i(\phi)} \quad \text{and} \quad \frac{\partial^2 \ell}{\partial \theta_i^2} = -\frac{b''(\theta_i)}{a_i(\phi)}.$$

The mean of the distribution, $\boldsymbol{\mu}$, can be deduced from the relation

$$\mathbb{E}\left[\frac{\partial \ell}{\partial \theta_i} \right] = 0 \implies \frac{\mu_i - b'(\theta_i)}{a_i(\phi)} = 0,$$

such that

$$\mathbb{E}[Y_i] = \mu_i = b'(\theta_i).$$

Similarly,

$$\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta_i^2}\right] + \mathbb{E}\left[\frac{\partial \ell}{\partial \theta_i}\right]^2 = 0 \implies -\frac{b''(\theta_i)}{a_i(\phi)} + \frac{\mathrm{Var}(Y_i)}{a_i(\phi)^2} = 0,$$

such that

$$\mathrm{Var}(Y_i) = b''(\theta_i) a_i(\phi).$$

6.2.2 Multi-view GKMs

In what follows, we build from the SRM objective function of a vector-valued kernel machine (5.1) and add an additional arbitrary regularisation term for the weight vector $\boldsymbol{w}$:

$$J(\boldsymbol{b}, f, \boldsymbol{w}) = \sum_{i=1}^{n} V\left(y_i, \boldsymbol{w}^\top \left(\boldsymbol{b} + f(\boldsymbol{x}_i)\right)\right) + \lambda \|f\|_{\mathcal{H}_K}^2 + \gamma R(\boldsymbol{w}),$$

where $V(\cdot)$ is the loss function. For a given learning task, if we assume that the response is generated according to an exponential family distribution (see Section 6.4 for examples covering the most common learning tasks), then the optimisation problem can be solved by maximum likelihood estimation. We can replace the loss function V with the negative log-likelihood of the exponential family distribution (assuming the canonical link, and omitting elements independent of $\boldsymbol{\eta}$):

$$J(\boldsymbol{b}, f, \boldsymbol{w}) = -\sum_{i=1}^{n} \left(y_i \eta_i - b(\eta_i)\right) + \lambda \|f\|_{\mathcal{H}_K}^2 + \gamma R(\boldsymbol{w}).$$

The Representer Theorem indicates that the minimiser $\widehat{f}$ is given by (5.2), so we can write

$$\boldsymbol{\eta} = (\boldsymbol{w}^\top \otimes \boldsymbol{I}_n)(\boldsymbol{b} \otimes \mathbf{1}_n + \boldsymbol{K}\boldsymbol{c}), \tag{6.1}$$

and

$$J(\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{w}) = -\boldsymbol{y}^\top \boldsymbol{\eta} + \mathbf{1}_n^\top b(\boldsymbol{\eta}) + \lambda \boldsymbol{c}^\top \boldsymbol{K}\boldsymbol{c} + \gamma R(\boldsymbol{w}). \tag{6.2}$$

6.2.3 Parameter Estimation

The optimisation strategy we propose is a coordinate descent scheme (Friedman et al., 2010), alternating between optimising $(\boldsymbol{b}, \boldsymbol{c})$ for a fixed $\boldsymbol{w}$ and vice versa. We describe the optimisation for each.

Optimising $(\boldsymbol{b}, \boldsymbol{c})$

As in Section 5.4, we can represent (6.2) in terms of $\boldsymbol{\alpha} \coloneqq (\boldsymbol{b}, \boldsymbol{c})$ by defining $\boldsymbol{V} \coloneqq (\mathbf{1}_n \otimes \boldsymbol{I}_v, \boldsymbol{K}) \in \mathbb{R}^{nv \times (n+1)v}$, $\boldsymbol{A} \coloneqq (\boldsymbol{w}^\top \otimes \boldsymbol{I}_n)\boldsymbol{V} \in \mathbb{R}^{n \times (n+1)v}$ and

$$\boldsymbol{U} \coloneqq \begin{bmatrix} \mathbf{0}_{v \times v} & \mathbf{0}_{v \times nv} \\ \mathbf{0}_{nv \times v} & \boldsymbol{K} \end{bmatrix},$$

yielding:

$$\boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{\alpha};$$

$$J(\boldsymbol{\alpha}, \boldsymbol{w}) = -\boldsymbol{y}^\top \boldsymbol{\eta} + \mathbf{1}_n^\top b(\boldsymbol{\eta}) + \lambda \boldsymbol{\alpha}^\top \boldsymbol{U} \boldsymbol{\alpha} + \gamma R(\boldsymbol{w}).$$

Following the same approach as before, we arrive at the Newton-Raphson update step:

$$\widehat{\boldsymbol{\alpha}}_{t+1} = \left[\boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{A} + 2\lambda \boldsymbol{U}\right]^{-1} \boldsymbol{A}^\top \boldsymbol{W}_t \boldsymbol{z}_t,$$

where $\boldsymbol{z}_t = \boldsymbol{A}\widehat{\boldsymbol{\alpha}}_t + \boldsymbol{W}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$ and $\boldsymbol{W} = \operatorname{diag}(b''(\eta_1), \ldots, b''(\eta_n))$.

Optimising $\boldsymbol{w}$

For the Representer Theorem to hold, we require that the regularisation term $\|f\|^2_{\mathcal{H}_K}$ be a squared L^2-norm. The same restriction is not required of the regularisation term on $\boldsymbol{w}$. For a fixed $\boldsymbol{\alpha}$, many optimisation procedures may be considered, depending on the properties of the chosen $R(\boldsymbol{w})$. Here we derive a Newton-Raphson update step that can be used for convex, differentiable $R(\boldsymbol{w})$ and show how it can be used to achieve L^2- and approximate L^1-regularisation.

We use the substitution $\boldsymbol{\eta} = (\boldsymbol{w}^\top \otimes \boldsymbol{I}_n)\boldsymbol{V}\boldsymbol{\alpha} \equiv \boldsymbol{B}\boldsymbol{w}$, where $\boldsymbol{B} \in \mathbb{R}^{n \times v}$ is a

matrix filled column-wise from the elements of $\boldsymbol{V\alpha} \in \mathbb{R}^{nv}$. The gradient and Hessian of (6.2) are then given by:

$$\nabla_{\boldsymbol{w}} = -\boldsymbol{B}^\top[\boldsymbol{y} - \boldsymbol{\mu}] + 2\gamma\nabla_{\boldsymbol{w}} R(\boldsymbol{w});$$

$$\nabla^2_{\boldsymbol{w}} = \boldsymbol{B}^\top \boldsymbol{W} \boldsymbol{B} + 2\gamma\nabla^2_{\boldsymbol{w}} R(\boldsymbol{w}),$$

where $\boldsymbol{W}$ is as before. The Newton-Raphson update is then:

$$\widehat{\boldsymbol{w}}_{t+1} = \widehat{\boldsymbol{w}}_t + \left[\boldsymbol{B}^\top \boldsymbol{W}_t \boldsymbol{B} + 2\gamma\nabla^2_{\boldsymbol{w}} R(\widehat{\boldsymbol{w}}_t)\right]^{-1} \left[\boldsymbol{B}^\top[\boldsymbol{y} - \boldsymbol{\mu}_t] - 2\gamma\nabla_{\boldsymbol{w}} R(\widehat{\boldsymbol{w}}_t)\right].$$

L^2-Regularisation

As the squared L^2-norm $R(\boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{w} \rangle$ is convex and twice differentiable, we can plug the following gradient and Hessian directly into the Newton-Raphson update formula:

$$\nabla_{\boldsymbol{w}} R(\boldsymbol{w}) = \boldsymbol{w}$$
$$\nabla^2_{\boldsymbol{w}} R(\boldsymbol{w}) = \boldsymbol{I}_v$$

to give:

$$\widehat{\boldsymbol{w}}_{t+1} = \widehat{\boldsymbol{w}}_t + \left[\boldsymbol{B}^\top \boldsymbol{W}_t \boldsymbol{B} + 2\gamma \boldsymbol{I}_v\right]^{-1} \left[\boldsymbol{B}^\top[\boldsymbol{y} - \boldsymbol{\mu}_t] - 2\gamma\widehat{\boldsymbol{w}}_t\right],$$

which can, as before, be re-written as

$$\widehat{\boldsymbol{w}}_{t+1} = \left[\boldsymbol{B}^\top \boldsymbol{W}_t \boldsymbol{B} + 2\gamma \boldsymbol{I}_v\right]^{-1} \boldsymbol{B}^\top \boldsymbol{W}_t \boldsymbol{u}_t,$$

where $\boldsymbol{u}_t = \boldsymbol{B}\widehat{\boldsymbol{w}}_t + \boldsymbol{W}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$.

L^1-Regularisation

If there are many views, some of which may be redundant, we often prefer to use an L^1-norm as a regularisation term:

$$R(\boldsymbol{w}) = \sum_{s=1}^{v} |w_s|.$$

Unfortunately, the L^1-norm is non-differentiable, and so first-order methods are typically used to solve such regularisation problems. Alternatively, we can replace the L^1-norm by a second-order differentiable parameterised function that approximates it. This has the advantage of allowing us to compute descent directions. The pseudo-Huber function (Hartley and Zisserman, 2003) approximates the L^1-norm for small values of $\xi > 0$ and has derivatives of all degrees.

$$R_\xi(\boldsymbol{w}) = \xi \sum_{s=1}^{v} \left(\sqrt{1 + \frac{w_s^2}{\xi^2}} - 1 \right)$$

The gradient of the pseudo-Huber function is given by

$$\nabla_{\boldsymbol{w}} R_\xi(\boldsymbol{w}) = \frac{1}{\xi} \left[w_1 \left(1 + \frac{w_1^2}{\xi^2} \right)^{-\frac{1}{2}}, \ldots, w_v \left(1 + \frac{w_v^2}{\xi^2} \right)^{-\frac{1}{2}} \right] \tag{6.3}$$

and the Hessian is given by

$$\nabla^2 R_\xi(\boldsymbol{w}) = \operatorname{diag} \left(\frac{1}{\xi} \left[\left(1 + \frac{w_1^2}{\xi^2} \right)^{-\frac{3}{2}}, \ldots, \left(1 + \frac{w_v^2}{\xi^2} \right)^{-\frac{3}{2}} \right] \right). \tag{6.4}$$

These can be plugged into the Newton-Raphson update formula, with ξ chosen sufficiently small to approximate L^1-shrinkage.

6.2.4 Algorithm

The algorithm iterates between updating $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ until some convergence criterion is met. A common stopping criterion is to measure the change in deviance between iterations. The deviance is defined through the difference of the log-likelihoods between the fitted model, $l(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{w}})$, and the saturated model, l_{sat}. If the canonical link function is used, computing l_{sat} amounts to setting $\theta_i = h(y_i)$. The deviance is then defined as:

$$D = -2[l(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{w}}) - l_{sat}]\phi,$$

which can be expressed as

$$D = 2\phi \sum_{i=1}^{n} \frac{y_i \left[h(y_i) - \eta_i\right] - b(h(y_i))}{a_i(\phi)} + b(\eta_i).$$

In most cases, $a_i(\phi) \propto \phi$, so the deviance does not depend on ϕ. Moreover, as our stopping criterion is based on a difference in deviances between iterations, we can ignore the terms involving ϕ, giving rise to the following expression:

$$D_t = \boldsymbol{y}^\top [h(\boldsymbol{y}) - \boldsymbol{\eta}_t] - b(h(\boldsymbol{y})) + b(\boldsymbol{\eta}_t)$$

The pseudocode for the algorithm is presented in Algorithm 2.

6.2.5 Prediction

Upon convergence, the estimated parameters $\widehat{\boldsymbol{\alpha}}$ and $\widehat{\boldsymbol{w}}$ can be used to make predictions on a set of test examples $\{\boldsymbol{x}_i^*\}_{i=1}^m$. Let $\boldsymbol{K}^* \in \mathbb{R}^{mv \times nv}$ be the kernel Gram matrix constructed by the evaluation of the operator-valued kernel between each test example and all labelled observations. We can then predict the values of the mean of the conditional distribution $\mu_i^* = \mathbb{E}(y_i \mid \boldsymbol{x}_i^*)$. The vector of predictions is given by:

$$\boldsymbol{\mu}^* = h^{-1}\left((\widehat{\boldsymbol{w}}^\top \otimes \boldsymbol{I}_m)\boldsymbol{V}^* \widehat{\boldsymbol{\alpha}}\right),$$

where $\boldsymbol{V}^* = (\boldsymbol{1}_m \otimes \boldsymbol{I}_v, \boldsymbol{K}^*) \in \mathbb{R}^{mv \times (n+1)v}$.

6.2.6 Outlier Detection

By choosing to adopt the GKM framework for multi-view learning, we have gained great flexibility to model any number of response types. Another advantage of this framework is the rich toolbox of diagnostic techniques that have been developed for traditional GLM models, which can be readily adapted for the kernelised multi-view analogues developed here. One particularly attractive diagnostic tool is Cook's distance (Cook, 1977), a technique for identifying samples with high *influence*, i.e. observations that have an

Algorithm 2: Multi-view Generalised Kernel Machine

Input:

$\boldsymbol{y} \in \mathcal{Y}^n$: vector of responses
$\boldsymbol{K} \in \mathbb{R}^{nv \times nv}$: $\mathcal{L}(\mathbb{R}^v)$-valued kernel Gram matrix
$\lambda, \gamma \in \mathbb{R}$: regularisation parameters
`model` $\in \{$"Gaussian", "logistic", "Poisson", $\dots\}$
`w_update` $\in \{$"L1", "L2"$\}$
`max_iter` $\in \mathbb{N}$: (optional) maximum number of iterations
$\epsilon \in \mathbb{R}$: convergence threshold

Initialise:

$\widehat{\boldsymbol{w}}_0 \leftarrow \frac{1}{v}\mathbf{1}_v$
$\widehat{\boldsymbol{\alpha}}_0 \leftarrow \mathbf{0}_{(n+1)v}$
$D_0 \leftarrow \infty$

Procedure:

Construct $\boldsymbol{V}$ and $\boldsymbol{U}$ from $\boldsymbol{K}$ as in Section 6.2.3
Choose $b(\cdot)$ and $h(\cdot)$ according to `model` (See Section 6.4)
for $t = 1$ *to* `max_iter` **do**
  $\boldsymbol{\eta}_t = (\boldsymbol{w}_{t-1}^\top \otimes \boldsymbol{I}_n)\boldsymbol{V}\widehat{\boldsymbol{\alpha}}_{t-1}$
  $D_t = \boldsymbol{y}^\top[h(\boldsymbol{y}) - \boldsymbol{\eta}_t] - b(h(\boldsymbol{y})) + b(\boldsymbol{\eta}_t)$
  if $|D_t - D_{t-1}| < \epsilon$ **then**
    break
  $\boldsymbol{\mu}_t = b'(\boldsymbol{\eta}_t)$
  $\boldsymbol{W}_t = \mathrm{diag}(b''(\boldsymbol{\eta}_t))$
  $\boldsymbol{z}_t = (\boldsymbol{w}_{t-1}^\top \otimes \boldsymbol{I}_n)\boldsymbol{V}\widehat{\boldsymbol{\alpha}}_{t-1} + \boldsymbol{W}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$
  $\widehat{\boldsymbol{\alpha}}_t \leftarrow$
    $\left[\boldsymbol{V}^\top(\boldsymbol{w}_{t-1} \otimes \boldsymbol{I}_n)\boldsymbol{W}_t(\boldsymbol{w}_{t-1}^\top \otimes \boldsymbol{I}_n)\boldsymbol{V} + \lambda\boldsymbol{U}\right]^{-1}\boldsymbol{V}^\top(\boldsymbol{w}_{t-1} \otimes \boldsymbol{I}_n)\boldsymbol{W}_t\boldsymbol{z}_t$
  Construct $\boldsymbol{B} \in \mathbb{R}^{n \times v}$ column-wise from $\boldsymbol{V}\widehat{\boldsymbol{\alpha}}_t \in \mathbb{R}^{nv}$
  $\boldsymbol{\eta}_t = \boldsymbol{B}\widehat{\boldsymbol{w}}_{t-1}$
  $\boldsymbol{\mu}_t = b'(\boldsymbol{\eta}_t)$
  $\boldsymbol{W}_t = \mathrm{diag}(b''(\boldsymbol{\eta}_t))$
  if `w_update` = L2 **then**
    $\boldsymbol{u}_t = \boldsymbol{B}\widehat{\boldsymbol{w}}_{t-1} + \boldsymbol{W}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$
    $\widehat{\boldsymbol{w}}_t \leftarrow \left[\boldsymbol{B}^\top\boldsymbol{W}_t\boldsymbol{B} + \gamma\boldsymbol{I}_v\right]^{-1}\boldsymbol{B}^\top\boldsymbol{W}_t\boldsymbol{u}_t$
  else if `w_update` = L1 **then**
    $\nabla_{\boldsymbol{w}} R_\mu(\widehat{\boldsymbol{w}}_{t-1}) =$ (6.3)
    $\nabla^2_{\boldsymbol{w}} R_\mu(\widehat{\boldsymbol{w}}_{t-1}) =$ (6.4)
    $\widehat{\boldsymbol{w}}_t \leftarrow \widehat{\boldsymbol{w}}_{t-1} +$
      $\left[\boldsymbol{B}^\top\boldsymbol{W}_t\boldsymbol{B} + \gamma\nabla^2_{\boldsymbol{w}} R(\widehat{\boldsymbol{w}}_{t-1})\right]^{-1}\left[\boldsymbol{B}^\top[\boldsymbol{y} - \boldsymbol{\mu}_t] - \gamma\nabla_{\boldsymbol{w}} R(\widehat{\boldsymbol{w}}_{t-1})\right]$

Output:

Estimated parameters $\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{w}}$

unusually strong impact on the fit of the regression model (and hence also on its downstream predictions).

Cook's distance was originally developed for ordinary least squares regression. Pregibon (1981) developed an approximation for Cook's distance in the case of GLMs. We will follow the same approach as Pregibon (1981) in deriving the Cook's distance score for multi-view GKMs.

Cook's Distance for Squared Error

Consider the maximum likelihood estimate (MLE) of $\boldsymbol{\alpha}$ that would be obtained if we used the least squares loss function, i.e. the solution of the objective function:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{(n+1)v}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\alpha}\|^2 + \lambda \boldsymbol{\alpha}^\top \boldsymbol{U} \boldsymbol{\alpha} + \gamma R(\boldsymbol{w}).$$

Setting the derivative w.r.t. $\boldsymbol{\alpha}$ equal to 0, we obtain the MLE:

$$\begin{aligned} \mathbf{0} &= -2\boldsymbol{A}^\top(\boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{\alpha}}) + 2\lambda \boldsymbol{U}\widehat{\boldsymbol{\alpha}} \\ \widehat{\boldsymbol{\alpha}} &= \left[\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{U}\right]^{-1} \boldsymbol{A}^\top \boldsymbol{y}. \end{aligned} \tag{6.5}$$

Cook's distance depends on the concept of *leverage* (see e.g. Chatterjee and Hadi (1986)). A high-leverage point is an observation made at an extreme or outlying region of the feature space, such that the lack of neighbouring observations causes the fitted regression model to pass close to that particular observation. Leverage scores are computed from the projection or "hat" matrix; that is, the matrix $\boldsymbol{H}$ that maps the vector of response values to the vector of fitted values: $\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$. In this case, the vector of fitted values is given by

$$\begin{aligned} \widehat{\boldsymbol{y}} &= \boldsymbol{A}\widehat{\boldsymbol{\alpha}} \\ &= \boldsymbol{A}\left[\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{U}\right]^{-1} \boldsymbol{A}^\top \boldsymbol{y}. \end{aligned}$$

and so the hat matrix is

$$\boldsymbol{H} = \boldsymbol{A}\left[\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{U}\right]^{-1}\boldsymbol{A}^\top.$$

The leverage for the ith sample is given by $h_i \coloneqq \boldsymbol{H}_{ii}$, the ith diagonal element of the hat matrix. Cook's distance can then be computed as

$$c_i = \widehat{r}_i^2 \frac{h_i}{(1-h_i)},$$

where $\widehat{r}_i$ is the ith element of the vector of residuals $\widehat{\boldsymbol{r}} = \boldsymbol{y} - \widehat{\boldsymbol{y}}$.

Cook's Distance for Multi-view GKMs

The parameters $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ were updated in Section 6.2 using an iteratively re-weighted least squares method. Upon convergence (where we have obtained $\boldsymbol{W} \coloneqq \lim_{t\to\infty}\boldsymbol{W}_t$), the maximum likelihood estimator for $\boldsymbol{\alpha}$ can be expressed as the solution of a weighted least squares problem:

$$\widehat{\boldsymbol{\alpha}} = \left[\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A} + \lambda\boldsymbol{U}\right]^{-1}\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{z}, \tag{6.6}$$

Comparing (6.6) with (6.5), we see that the same regression weights would be obtained if we had performed an ordinary linear regression of *pseudo-observations* $\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{z}$ on *pseudo-predictors* $\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{A}$. Under this representation, all model diagnostics from linear regression are readily applied. The hat matrix becomes:

$$\boldsymbol{H}^{(\boldsymbol{\alpha})} = \boldsymbol{W}^{\frac{1}{2}}\boldsymbol{A}\left[\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A} + \lambda\boldsymbol{U}\right]^{-1}\boldsymbol{A}^\top\boldsymbol{W}^{\frac{1}{2}}$$

and the generalised version of Cook's distance becomes:

$$c_i^{(\boldsymbol{\alpha})} = (\widetilde{r}_i)^2 \frac{h_i}{(1-h_i)}, \tag{6.7}$$

where $\widetilde{\boldsymbol{r}} = \boldsymbol{W}^{\frac{1}{2}}\widehat{\boldsymbol{r}}$.

Note that (6.7) measures outliers with respect to the estimation of $\boldsymbol{\alpha}$. We

can follow the same procedure to estimate Cook's distances with respect to the estimation of $\boldsymbol{w}$, yielding the following hat matrix:

$$\boldsymbol{H}^{(\boldsymbol{w})} = \boldsymbol{W}^{\frac{1}{2}}\boldsymbol{B}\left[\boldsymbol{B}^{\top}\boldsymbol{W}\boldsymbol{B} + \gamma\boldsymbol{I}_n\right]^{-1}\boldsymbol{B}^{\top}\boldsymbol{W}^{\frac{1}{2}}.$$

Intuitively, a high value of $h_i^{(\boldsymbol{\alpha})}$ suggests that an observation lies in an outlying region of the (possibly infinite dimensional) reproducing kernel Hilbert space implied by the multi-view kernel. A high value of $h_i^{(\boldsymbol{w})}$ identifies observations that are outlying in $\mathbb{R}^v$.

6.3 Multi-view Kernel Functions

We have described an algorithm for learning from multi-view data. Information from within and between view representations is encoded into the algorithm through the operator-valued kernel Gram matrix. In this section, we describe three ways to construct a positive-semidefinite operator-valued kernel using individual scalar-valued kernel functions $k_s(\cdot,\cdot)$ for each view $s \in \{1, ..., v\}$. The kernel functions do not need to be of the same type and can be chosen appropriate to the data type of each view. Some common choices include: for vectorial data—the radial basis function kernel, polynomial kernel and Laplacian kernel; for variable length strings—the substring kernel (Hastie et al., 2009, p. 668); for graphs—the path kernel (Deng et al., 2012, p. 99); for histograms—the generalised histogram intersection kernel (Boughorbel et al., 2005). For a survey of kernels for structured data types, see Gärtner (2003).

Given v scalar-valued kernels, we can then construct a class of $\mathcal{L}(\mathbb{R}^v)$-valued kernels, which takes the following form:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j)_{lm} = \left\langle \phi_l\left(\boldsymbol{x}_i^{(l)}\right), \Sigma_{lm}\phi_m\left(\boldsymbol{x}_j^{(m)}\right)\right\rangle_{\mathcal{H}_{k_l}}, \tag{6.8}$$

where $\phi_s(\cdot)$ is the feature map associated with the kernel k_s. The linear operator Σ_{lm} allows us to incorporate within- and between-view information.

6.3.1 MKL Kernel

In the simplest case, we define

$$\Sigma_{lm} = \begin{cases} 1 \text{ if } l = m \\ \theta_{lm} \text{ otherwise,} \end{cases}$$

where $\theta_{lm} \in \mathcal{L}(\mathcal{H}_{k_m}, \mathcal{H}_{k_l})$ is the zero map, $\theta_{lm} : \phi_m\left(\boldsymbol{x}^{(m)}\right) \mapsto \mathbf{0} \in \mathcal{H}_{k_l}$. The multi-view kernel then reduces to

$$K_{\text{MKL}}(\boldsymbol{x}_i, \boldsymbol{x}_j)_{lm} = \begin{cases} k_l(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ if } l = m, \\ 0 \text{ otherwise,} \end{cases} \tag{6.9}$$

which was the kernel we used in Chapter 5. The Gram matrix $\boldsymbol{K}_{\text{MKL}}$ is block-diagonal and each block corresponds to the scalar-valued kernel Gram matrix over one view. Each element of the learned function $f = (f^{(1)}, \ldots, f^{(v)})$ thus uses within-view information only and the optimal weighting of these functions is learned through $\boldsymbol{w}$.

This has a clear resemblance to Multiple Kernel Learning (MKL) algorithms. A challenge that these types of algorithms face is constraining the learned weights to be non-negative. This is required in order to ensure the positive-definiteness of the combined kernel. In our case, the multi-view kernel (6.9) is by definition positive-semidefinite, as the Gram matrix $\boldsymbol{K}_{\text{MKL}}$ is block-diagonal with each block being positive-semidefinite. Our learning scheme therefore provides a simple and elegant optimisation procedure for MKL. This again highlights the benefit of our approach over that of Minh et al. (2016), who cannot find an optimal weighting of kernels in the classification setting.

6.3.2 Cross-covariance Kernel

Cross covariance operators were first used for solving problems in information theory (Baker, 1970) but have more recently received attention by researchers in machine learning, who have employed them for measuring statistical dependence (Gretton et al., 2005) and for dimensionality reduction (Fukumizu

et al., 2004, 2009). To our knowledge, Kadri et al. (2013) is the only work to use a multi-view kernel based on cross-covariance operators. The cross-covariance operator is defined as:

$$\boldsymbol{\Sigma}_{lm} = \mathbb{E}\left[\phi_l(X_l) \otimes \phi_m(X_m)\right] - \mathbb{E}\left[\phi_l(X_l)\right] \otimes \mathbb{E}\left[\phi_m(X_m)\right].$$

In order to compute the kernel (6.8), we must estimate the cross-covariance operator from the training data. Let $\Phi_s = (\phi_s(\boldsymbol{x}_1), \ldots, \phi_s(\boldsymbol{x}_n))$ denote a matrix of the images of the training data under ϕ_s and let $\hat{\mu}_{X_s} = \frac{1}{n}\sum_{i=1}^{n} \phi_s(\boldsymbol{x}_i)$ be the empirical estimate of the expected feature map $\mathbb{E}[\phi_s(X_s)]$. The empirical cross-covariance operator is then given by:

$$\begin{aligned}
\widehat{\boldsymbol{\Sigma}}_{lm} &= \frac{1}{n}\left(\Phi_l - \hat{\mu}_{X_l}\mathbf{1}_n^\top\right)\left(\Phi_m - \hat{\mu}_{X_m}\mathbf{1}_n^\top\right)^\top \\
&= \frac{1}{n}\left(\Phi_l - \frac{1}{n}\Phi_l\mathbf{1}_n\mathbf{1}_n^\top\right)\left(\Phi_m - \frac{1}{n}\Phi_m\mathbf{1}_n\mathbf{1}_n^\top\right)^\top \\
&= \frac{1}{n}\Phi_l\boldsymbol{H}\Phi_m^\top,
\end{aligned}$$

where $\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ is a centring matrix.

The multi-view kernel (6.8) can thus be constructed as:

$$\begin{aligned}
K_{\text{CC}}(\boldsymbol{x}_i, \boldsymbol{x}_j)_{lm} &= \left\langle \phi_l(\boldsymbol{x}_i), \frac{1}{n}\Phi_l\boldsymbol{H}\Phi_m^\top\phi_m(\boldsymbol{x}_j)\right\rangle_{\mathcal{H}_l} \\
&= \frac{1}{n}\left\langle \Phi_l^\top\phi_l(\boldsymbol{x}_i), \boldsymbol{H}\Phi_m^\top\phi_m(\boldsymbol{x}_j)\right\rangle \\
&= \frac{1}{n}\left\langle \boldsymbol{k}_l(\boldsymbol{x}_i), \boldsymbol{H}\boldsymbol{k}_m(\boldsymbol{x}_j)\right\rangle,
\end{aligned}$$

where we have written $\boldsymbol{k}_s(\boldsymbol{x}_i) = (k_s(\boldsymbol{x}_i, \boldsymbol{x}_t))_{t=1}^n$, i.e. the ith column of the Gram matrix $\boldsymbol{K}_s$. It is then easy to see that the Gram matrix $\boldsymbol{K}_{\text{CC}}$ corresponding to this multi-view kernel $K_{\text{CC}}(\cdot,\cdot)$ is a block matrix, with the lm-th block equal to $\frac{1}{n}\boldsymbol{K}_l\boldsymbol{H}\boldsymbol{K}_m$.

6.3.3 Learned Metric Kernel

Huusari et al. (2019) consider linear operators Σ_{lm} that can be expressed

as $\Sigma_{lm} = \Phi_l \boldsymbol{M}_{lm} \Phi_m^\top$, where $\boldsymbol{M}_{lm}$ is a positive definite matrix that plays the role of a metric between the two features maps associated with kernels k_l and k_m. The multi-view kernel is then given by $K_{\text{MVML}}(\boldsymbol{x}_i, \boldsymbol{x}_j)_{lm} = \langle \boldsymbol{k}_m(\boldsymbol{x}_i), \boldsymbol{M}_{lm} \boldsymbol{k}_m(\boldsymbol{x}_j) \rangle$, and it is easy to see that the block kernel Gram matrix $\boldsymbol{K}_{\text{MVML}}$ corresponding to this multi-view kernel is given by

$$\boldsymbol{K}_{\text{MVML}} = \boldsymbol{K}_{\text{MKL}} \boldsymbol{M} \boldsymbol{K}_{\text{MKL}},$$

where $\boldsymbol{M} = (\boldsymbol{M}_{lm})_{l,m=1}^{v} \in \mathbb{R}^{nv \times nv}$ encodes pairwise similarities between all of the views.

Multi-view Metric Learning (MVML), as proposed by Huusari et al. (2019), corresponds to simultaneously learning the metric $\boldsymbol{M}$ and the classifier or regressor. This is achieved through the inclusion of an additional regularisation term into the objective function, which adds a further step to the alternating gradient descent optimisation scheme. For the sake of clarity in our exposition of the optimisation strategy presented in Section 6.2, we have not considered the simultaneous learning of the kernel metric $\boldsymbol{M}$ in our framework.

6.4 Example Response Types

We now provide some examples of exponential family distributions that can be used to model different response types that are frequently encountered in real-world learning problems.

6.4.1 Logistic Regression

With binary response data, the target y_i indicates whether the ith observation belongs to the positive ($y_i = 1$) or negative ($y_i = 0$) class. In this case, the responses can be viewed as a series of Bernoulli trials, such that

$$f(y_i \ ; \ \pi_i) = \pi_i^{y_i} (1 - \pi_i)^{1 - y_i},$$

where π_i represents the probability (conditioned on the input $\boldsymbol{x}_i$) that the ith example belongs to the positive class. The Bernoulli distribution can be written as a one-parameter member of the exponential family as

$$f(y_i \; ; \; \theta_i) = \exp\left(y_i\theta_i - \log\left(1 + \exp(\theta_i)\right)\right),$$

where

$$\pi_i = \frac{\exp(\theta_i)}{1 + \exp(\theta_i)}.$$

In this case we have $b(\theta_i) = \log(1 + \exp(\theta_i))$. Taking the canonical link such that $\theta_i = \eta_i$, we have that

$$\eta_i = h(\pi_i) = \text{logit}(\pi_i) = \log\left(\frac{\pi_i}{1 + \pi_i}\right).$$

The parameters can be estimated as in Section 6.2, where $b''(\eta_i) = \pi_i(1 - \pi_i)$.

6.4.2 Poisson Regression

When the response data are counts, it is sensible to restrict the estimate of the conditional mean to be non-negative integers. One choice is the Poisson distribution, which models the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant rate and independently of the time since the last event. The Poisson distribution is given by:

$$\begin{aligned} f(y_i \; ; \; \mu_i) &= \frac{\exp(-\mu_i)\mu_i^{y_i}}{y!} \\ &= \exp\left(y_i\log(\mu_i) - \mu_i - \log(y_i!)\right), \end{aligned}$$

such that $\theta = \log(\mu)$ and $b(\theta) = \exp(\theta)$; hence the canonical link is given by

$$\eta_i = h(\mu_i) = \log(\mu_i).$$

For parameter estimation, $b''(\eta_i) = \exp(\eta_i)$.

6.4.3 Multinomial Logistic Regression

Often in classification problems, there are more than two possible discrete outcomes. We can represent this scenario using a vector response $\boldsymbol{y}_i = (y_{i,1}, \ldots, y_{i,g})^\top$, where $y_{i,m} \in \{0, 1\},\ m = 1, \ldots, g$ and $\sum_{m=1}^{g} y_{i,m} = 1$. The probability of observing $\boldsymbol{y}_i$ is then given by the categorical distribution:

$$f(\boldsymbol{y}_i \ ; \ \boldsymbol{\pi}_i) = \prod_{m=1}^{g} \pi_{i,m}^{y_{i,m}}.$$

where $\boldsymbol{\pi}_i = (\pi_{i,1}, \ldots, \pi_{i,g})^\top$ represents the probability of each class. Due to the fact that $\pi_{i,g} = 1 - \sum_{m=1}^{g-1} \pi_{i,m}$, an equivalent parameterisation in terms of only the first $g - 1$ components permits an expression in the canonical form of a *multivariate* exponential family distribution:

$$f(\boldsymbol{y}_i \ ; \ \boldsymbol{\theta}_i) = \exp\left(\boldsymbol{y}_i^\top \boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)\right),$$

for $\boldsymbol{y}_i = (y_{i,1}, \ldots, y_{i,g-1})^\top$ and $\boldsymbol{\theta}_i = (\theta_{i,1}, \ldots, \theta_{i,g-1})^\top$, with

$$\theta_{i,m} = \log\left(\frac{\pi_{i,m}}{1 - \sum_{s=1}^{g-1} \pi_{i,s}}\right)$$

and

$$b(\boldsymbol{\theta}_i) = -\log\left(1 - \sum_{m=1}^{g-1} \exp\left(\theta_{i,m}\right)\right).$$

The optimisation procedure outlined in Section 6.2.3 is readily extended to this multivariate setting by simply expanding dimensionalities. As the response vector is now given by $\boldsymbol{y} = (\boldsymbol{y}_i, \ldots, \boldsymbol{y}_n) \in \mathbb{R}^{(g-1)n}$ and we wish to estimate $\boldsymbol{\alpha} \in \mathbb{R}^{(g-1)(n+1)v}$, we need to make a number of modifications to the components of (5.13). Expand $\boldsymbol{A} \in \mathbb{R}^{n \times (n+1)v}$ to make $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{(g-1)n \times (g-1)(n+1)v}$

as follows:

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{I}_{g-1} \otimes \boldsymbol{A}_{1,*} \\ \boldsymbol{I}_{g-1} \otimes \boldsymbol{A}_{2,*} \\ \vdots \\ \boldsymbol{I}_{g-1} \otimes \boldsymbol{A}_{n,*} \end{bmatrix},$$

where $\boldsymbol{A}_{i,*}$ denotes the ith row of $\boldsymbol{A}$. For the $\boldsymbol{W}$ analogue, let the block matrix $\boldsymbol{\mathcal{W}} \coloneqq \text{diag}(\boldsymbol{\mathcal{W}}_1, \ldots, \boldsymbol{\mathcal{W}}_n)$, with $\boldsymbol{\mathcal{W}}_i = \{\pi_{i,s}(\delta_{st} - \pi_{i,t})\}_{s,t=1}^{g-1} \in \mathbb{R}^{(g-1)\times(g-1)}$, where δ_{st} is the Kronecker delta function. Finally, let $\boldsymbol{\mathcal{U}} = \boldsymbol{I}_{g-1} \otimes \boldsymbol{U}$. The Newton-Raphson update step (5.13) is then given by:

$$\widehat{\boldsymbol{\alpha}}_{t+1} = \left[\boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{W}}_t \boldsymbol{\mathcal{A}} + \lambda \boldsymbol{\mathcal{U}}\right]^{-1} \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{W}}_t \boldsymbol{z}_t, \tag{6.10}$$

for $\boldsymbol{z}_t = \boldsymbol{\mathcal{A}}\widehat{\boldsymbol{\alpha}}_t + \boldsymbol{\mathcal{W}}_t^{-1}[\boldsymbol{y} - \boldsymbol{\mu}_t]$.

Multinomial Cook's Distance

The Cook's distance formulation presented in Section 6.2.6, based on Pregibon (1981), assumes a scalar-valued response. Lesaffre and Albert (1989) extend the work of Pregibon (1981) to the multinomial case. From (6.10), we see that in the multinomial case, the hat matrix is given by:

$$\boldsymbol{H}^{(\boldsymbol{\alpha})} = \boldsymbol{\mathcal{W}}^{\frac{1}{2}} \boldsymbol{\mathcal{A}} \left[\boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{W}} \boldsymbol{\mathcal{A}} + \lambda \boldsymbol{\mathcal{U}}\right]^{-1} \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{W}}^{\frac{1}{2}}.$$

which is a block-matrix, where each block $\boldsymbol{H}_{ij}^{(\boldsymbol{\alpha})} \in \mathbb{R}^{(g-1)\times(g-1)}$. Just as in the two-group case, the diagonal elements (here, blocks) can be used to define the concept of a leverage point. Let $\boldsymbol{G} = \boldsymbol{I} - \boldsymbol{H}$, then a small value of $|\boldsymbol{G}_{ii}| < 1$ indicates that observation i has high leverage, as its omission from the sample greatly increases the variability of the estimated coefficients—see Lesaffre and Albert (1989) for details. Observations with high influence can be identified by calculating an approximate Cook's distance:

$$c_i^{(\boldsymbol{\alpha})} = \boldsymbol{\chi}_i^\top \boldsymbol{G}_{ii}^{-1} \boldsymbol{H}_{ii} \boldsymbol{G}_{ii}^{-1} \boldsymbol{\chi}_i,$$

where $\boldsymbol{\chi} = \boldsymbol{\mathcal{W}}^{-\frac{1}{2}} \widehat{\boldsymbol{r}}$ is the standardised residual vector.

Despite the fact that GLMs are among the most widely used and foundational methods of statistical analysis, and Cook's distance is a core diagnostic tool thereof, the multinomial analogue presented by Lesaffre and Albert (1989) has received little attention or application. This can perhaps be explained by the disparity between the computational resources required for its calculation and the limited processing and storage capabilities available at the time of its introduction. Though this has since changed dramatically, multinomial Cook's distances remain, in our view, under-utilised. Hosmer Jr et al. (2013, p. 284) comment on their protracted obscurity, noting that "this is somewhat surprising given that programs to fit the multinomial logistic regression model are now widespread". We hope that our demonstration in the following section will motivate practitioners to consider multinomial Cook's distances in any future multi-class classification problems that they may face, whether multi-view or otherwise.

6.5 Experiments

In this section we investigate the performance of our Multi-View Generalised Kernel Machine (MVGKM) method for several realistic real-world applications. First, to showcase the flexibility of MVGKM, we apply it to a number of different learning tasks and compare its performance with early and late data fusion strategies. Following this, we demonstrate how we can leverage Cook's distances for anomaly detection in cyber security applications.

To facilitate adoption, experimentation and comparison among the multi-view learning community, we implement our method[1] to be compatible with the `scikit-multimodallearn` package of Benielli et al. (2022). This allows us to take advantage of the rich toolbox of functionalities available in `scikit-learn` (Pedregosa et al., 2011) for building machine learning pipelines, performing hyperparameter grid searches and cross-validation.

[1] `https://github.com/jackhogan/MVGKM`

6.5.1 Binary, Multi-class and Poisson Regression

To demonstrate the flexibility of MVGKM, we apply it to the following multi-view datasets, which contain a diverse selection of data types and corresponding feature representations, as well as distinct response types that match those described in the preceding section.

- **MVSA-Single:** This dataset,[2] constructed by Niu et al. (2016) contains 5,129 image-text pairs collected from posts on Twitter. Each pair has been annotated with a class label describing whether the post is positive, negative or neutral in its sentiment. For the purpose of our experiment, we treat this as a binary classification problem by considering only the 2,948 positive and negative examples. This is a natural 2-view dataset and we construct a single feature representation per view. For the image view, we extract GIST feature vectors (Oliva and Torralba, 2001). We represent the text view using a Bag-of-Words dictionary. 80% of the dataset was used for training and the remaining 20% was reserved as a test set.

- **uWave Gesture Library:** The uWave[3] dataset (Liu et al., 2009) contains measurements recorded using a handheld accelerometer while participants performed eight specific gestures. Though this has only a single modality, we can consider this as a 3-view dataset, represented using time series vectors of the accelerometer position along the X, Y and Z dimensions for the duration of the gesture motion. There are 896 training examples and 3,582 test examples.

- **LibriCount:** This synthetic dataset[4] was generated by Stöter et al. (2018) to mimic the so-called cocktail party problem, which occurs when sounds from multiple sources mix before arriving at the ear. It comprises 5,720 audio wave files of 5s duration; in each, random

[2] https://mcrlab.net/research/mvsa-sentiment-analysis-on-multi-view-social-data/

[3] https://www.timeseriesclassification.com/description.php?Dataset=UWaveGestureLibraryAll

[4] https://zenodo.org/record/1216072

	MVSA-Single (AUC)	uWave (Acc.)	LibriCount (MAE)
$\text{MVGKM}_{\text{MKL}}$	**82.99**	93.80	**0.66**
MVGKM_{CC}	78.94	67.87	0.77
MVML	–	**94.84**	1.02
$\text{SVM}_{\text{early}}$	78.94	94.42	0.82
SVM_{late}	74.62	82.74	0.96

Table 6.1: Experimental results. Note that MVML outputs only a binary prediction so was omitted from the MVSA-Single experiment.

utterances from varying numbers of speakers $(0 - 10)$ are mixed together. This is a single-modality dataset, though we can describe each instance using multiple feature spaces. We use the `librosa` Python library (McFee et al., 2015) to extract three feature representations: Mel-frequency cepstral coefficients (Mermelstein, 1976), root-mean-square values, and chromagram features (Ellis and Poliner, 2007). We randomly sample 1,000 instances for our training set and use the remainder as a test set.

Experimental Setup

For each learning task, we compare the efficacy of our proposed method against the two most common approaches to learning from multi-view data: so-called *early* and *late fusion* (as described in Section 2.2). We also include the Multi-View Metric Learning (MVML) method of Huusari et al. (2019) in our comparison, which is available as part of the `scikit-multimodallearn` package.

As MVGKM and MVML are kernel-based methods, we perform early and late fusion using support vector machines and refer to them as $\text{SVM}_{\text{early}}$ and SVM_{late}, respectively. For the Bag-of-Words text feature representation in the MVSA-Single dataset, we use a linear kernel. In all other cases, we use a Gaussian kernel

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{z}\|^2\right),$$

with σ chosen to be the mean of distances in the training set. For all experiments, we tune the hyperparameters of each classifier by performing a grid search on the training set over a range of values and using 5-fold cross-validation. The following scoring metrics for model selection and comparison were chosen: for MVSA-Single, we use area under the ROC curve (AUC); for uWave, we use multi-class accuracy; for LibriCount, we use mean absolute error (MAE).

It is important to emphasise that for all experiments, we expect that superior performance could be achieved through a number of modifications to the experimental design presented here. Domain expertise with text, audio or image data would inform a more considered approach to feature extraction, kernel design and selection. However, our goal here is not to outperform the state-of-the-art in any particular problem but to showcase how our method performs as well as common alternatives, while also providing a number of additional desirable features and advantages.

Results

Results from the three experiments are listed in Table 6.1.

In the MVSA-Single sentiment classification experiment, the MVGKM model with MKL kernel outperforms early and late fusion SVMs. Their ROC and Precision/Recall curves are plotted in Figure 6.1a. At almost all points, the MVGKM curve dominates the others. One advantage of (logistic) MVGKM for binary classification is that it is well calibrated by default. The support vector machine output requires calibration using Platt scaling (Platt, 1999) to produce probabilistic predictions. In Figure 6.1b, we plot density estimates of the predicted probabilities for $\text{MVGKM}_{\text{MKL}}$ and $\text{SVM}_{\text{early}}$ grouped by true class label. We can see that for negative Twitter posts, the predictions made by $\text{SVM}_{\text{early}}$ are nearly uniformly distributed.

For the multi-class classification experiment, MVML achieves the highest accuracy, with $\text{SVM}_{\text{early}}$ and $\text{MVGKM}_{\text{MKL}}$ marginally lower. It is interesting to note that in this case, there appears to be little advantage gained from

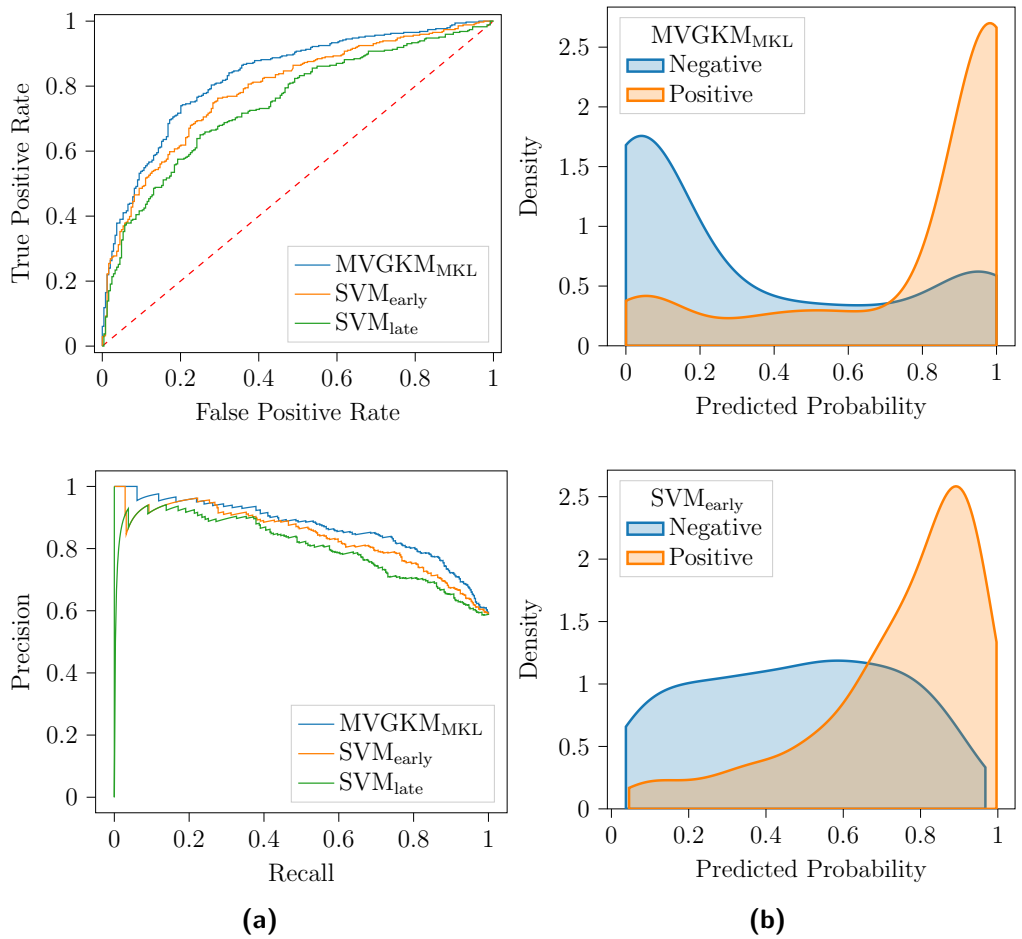


Figure 6.1: Results from the binary sentiment classification experiment. Column (a) shows Receiver Operation Characteristic curves and Precision/Recall curves for three classifiers. Column (b) shows kernel density estimates of the predicted probabilities made by the MVGKM$_{MKL}$ and SVM$_{early}$ classifiers, grouped by the true class label.

adopting a multi-view approach in favour of simply concatenating feature vectors. This can be explained by the fact that the three views are all of the same modality: spatial coordinates in the X, Y and Z dimensions. In the first experiment, one view was GIST features from an image—real-valued and dense—while the other view was a Bag-of-Words dictionary—integer-valued and sparse. MVML benefits from the ability to apply a custom kernel to each modality individually and learn a metric to incorporate between-view information. In Figure 6.2a, we take advantage of MVGKM's ability to compute multinomial Cook's distances in order to investigate whether the uWave

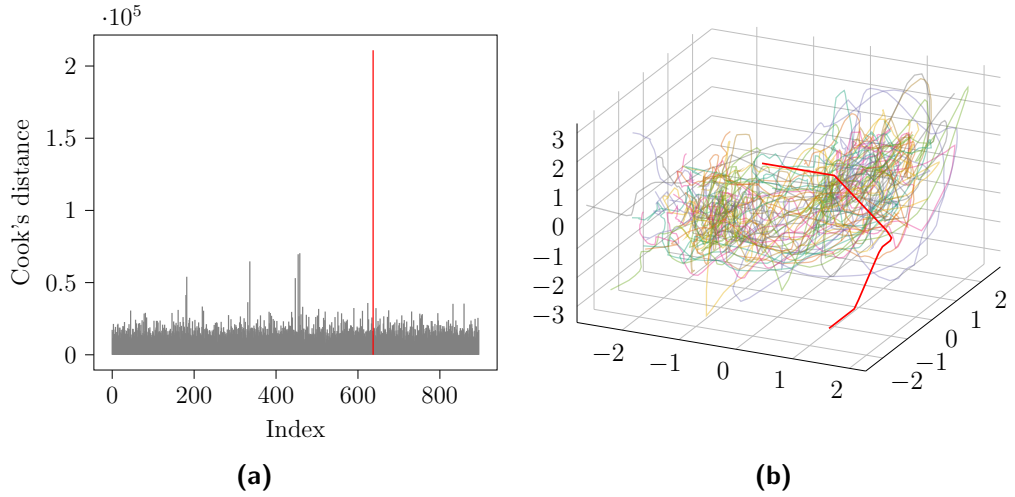


Figure 6.2: Illustration of outlier detection using MVGKM. (a) shows the multinomial Cook's distance for each observation in the uWave training set. (b) shows a sample of 50 observations plotted in 3-dimensions. All are of the same class as the outlier observation, which is plotted in red.

dataset contains any outliers. A large spike is immediately obvious. As the uWave dataset comprises paths in 3-dimensional space, we can visually investigate whether this observation does appear to differ substantially from others. In Figure 6.2b, its path is plotted in red against a sample of other observations from the same class; a departure from the pattern displayed by other observations is clear. One could argue that such an obvious outlier could be detected by other means as part of a pre-processing step prior to model training with an arbitrary learning scheme. However, for many learning problems, mixed modalities and high dimensionalities make the discovery of outliers extremely difficult without a built-in functionality like that of MVGKM.

In the LibriCount voice counting experiment, (Poisson) $\text{MVGKM}_{\text{MKL}}$ scored the lowest MAE on the test set. The learned model is naturally constrained to predict non-negative integer responses, in contrast with MVML and SVMs, which can predict negative values. Figure 6.3a shows the MAE of each classifier conditioned on the true value of the response. Figure 6.3b shows a normalised confusion matrix of the true response versus predicted number of voices for the MVGKM model.

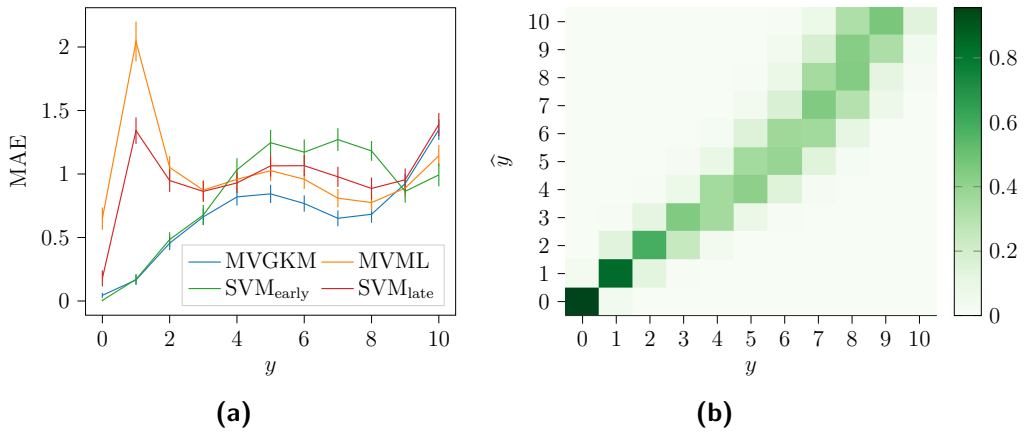


Figure 6.3: Results from the LibriCount voice counting experiment. (a) shows the mean absolute error (MAE) for each ground-truth value of the response. Error bars show the 95% confidence intervals. (b) shows a normalised confusion matrix of the predicted value $\widehat{y}$ versus the true response y for the best performing model (Poisson MVGKM).

6.5.2 Anomaly Detection

In Chapter 3, we proposed a framework for detecting anomalous activity based on a discrepancy between the observed value and its prediction. In effect, we were identifying points with large *residuals*. One of the strengths of the GKM framework we have since developed is the ability to detect points that have both large residuals *and/or high leverage* (see Section 6.2.6). By combining both these concepts, Cook's distance gives us a fuller picture of a given observation's 'unusualness'. In this experiment, we investigate the efficacy of applying our outlier detection framework to the LANL dataset.

Experimental Setup

As in Chapter 3, we model a computer's activity during fixed intervals of time. For this experiment, the data is divided into 15-minute bins. For a given computer, the response variable we choose to model is the number of unique destination ports used for communications during that bin. This is known to be a quantity of interest, as malicious actors often employ a technique known as "port scanning" before an attack, whereby requests are sent to hosts using a range of destination ports in order to learn details

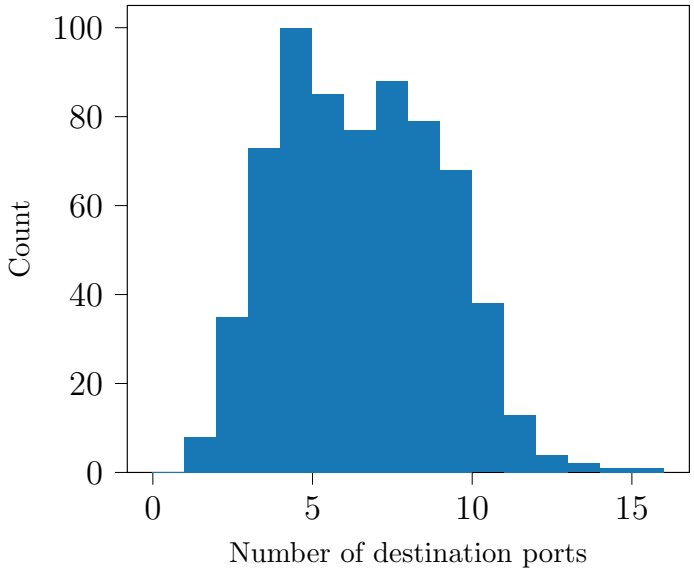


Figure 6.4: A histogram of the number of unique destination ports used during a 15-minute bin in communications by a specific computer from the LANL dataset (`Comp007792`) over 7 days.

about their running services or locate potential vulnerabilities (Panjwani et al., 2005). A device is chosen from the WLS dataset, to ensure that it is a computer with both NetFlow and process activity. A histogram of the value of the response variable over 7 days (i.e. 672 bins) is shown in Figure 6.4. As there are no counts of zero, this computer is active continuously over the duration of the experiment.

We consider two views of the data and choose appropriate kernel functions:

- **Netflow:** As described in Section 3.2, the Netflow data during a given time interval can be considered as a graph containing nodes C_i and edges $a_{i,j} \in \{0, 1\}$ indicating the presence/absence of a communication between C_i and C_j. For bin i, we construct $\boldsymbol{x}_i^{(1)} = \{C_j \ ; \ a_{i,j} = 1\}$, i.e. the set of devices with which our target computer communicates. For our kernel, we use the Jaccard index (Jaccard, 1912):

$$k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{|\boldsymbol{x}_i^{(1)} \cap \boldsymbol{x}_j^{(1)}|}{|\boldsymbol{x}_i^{(1)} \cup \boldsymbol{x}_j^{(1)}|}.$$

Gower (1971) proves that this is a valid positive-semidefinite kernel function.

- **Processes:** For bin i, we gather from the WLS dataset a list of all the processes that run on our target device. We use a tf-idf bag-of-words representation and the cosine kernel function:

$$k_2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^{(2)\top} \boldsymbol{x}_j^{(2)}}{\|\boldsymbol{x}_i^{(2)}\| \|\boldsymbol{x}_j^{(2)}\|}.$$

A Poisson $\text{MVGKM}_{\text{MKL}}$ regression model (Section 6.4.2) was trained using 7 days of LANL data. The hyperparameters λ and γ were tuned using 5-fold cross validation. The best performing combination achieved a mean MAE of 0.96 across the test folds.

Results

Once the hyperparameters were selected, the model was re-trained on the full 7 days of data. The Cook's distance values for each bin were calculated using (6.7); these are shown in Figure 6.5a. The value for bin 100 is shown in red, where $y_{100} = 6$. As an experiment, we changed the response value for that observation to be $y_{100} = 15$ and re-trained the model. When the Cook's distances are calculated again on the new model, a clear spike appears at index 100. This can be seen in Figure 6.5b. Note that we observed similar spikes when we changed the response for other bins, provided the change was large enough, i.e. from relatively low to high but not for smaller changes (e.g. from 10 to 12).

This is a promising result, indicating that the model is able to identify when the response for a given bin is anomalous with respect to its associated features. This is all the more impressive when we remind ourselves that these 'features' are actually a representation of the data in a possibly infinite-dimensional vector-valued Hilbert space implied by a multi-view kernel. In the following chapter, we develop a method of learning an explicit finite-dimensional representation of multi-view data.

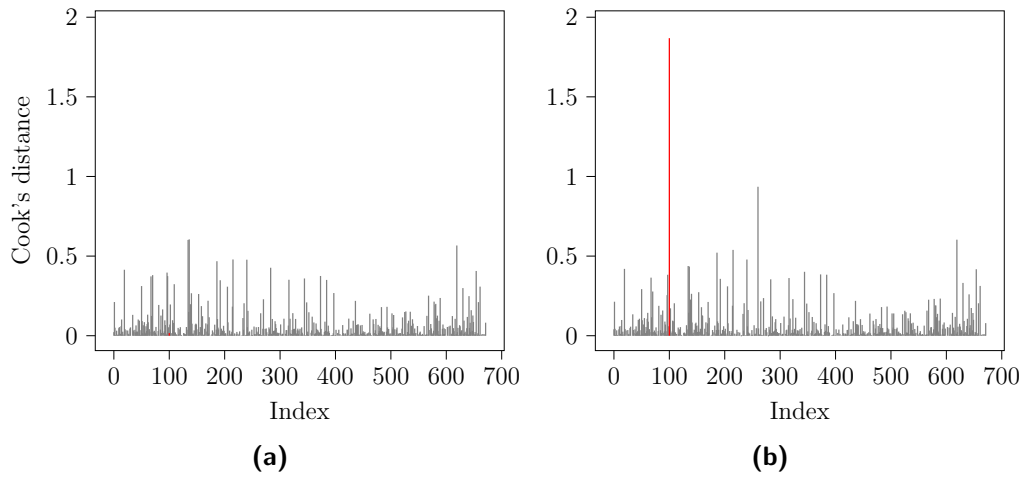


Figure 6.5: Cook's distance plots for 7 days (i.e. 672 bins) of LANL data, computed from a Poisson MVGKM$_{\text{MKL}}$ regression model that was trained using: (a) the true response values; (b) the response values with y_{100} changed from 6 to 15.

6.6 Discussion

In this chapter, we have presented a new general framework for multi-view supervised learning. We extended the Generalised Linear Model paradigm to operate in vector-valued RKHSs and, by doing so, developed a single unifying learning scheme for modelling an arbitrary response type using data that can be described using multiple, heterogeneous feature representations. Through experiments with real data, we demonstrated the flexibility and user-friendliness of our framework. Beyond what has been presented here, the framework can be further extended with additional link functions, multi-view kernels and diagnostic tools. Of particular interest would be the inclusion of the simultaneous metric learning capability of MVML, as in our experiments, the cross-covariance multi-view kernel performs poorly at incorporating between-view information. We have implemented MVGKM to be compatible with the `scikit-multimodallearn` library so that it is available for adoption and adaption by members of the multi-view learning community. Having considered semi-supervised and supervised learning, in the following chapter we develop a novel approach to multi-view unsupervised learning.

7

Multi-view Multidimensional Scaling

In Chapters 3 and 6, we presented examples of ways in which multi-view learning could be used for anomaly detection in cyber security. Recall that in both cases, anomalies were defined with respect to a supervised learning model's prediction of some pre-specified quantity of interest: in Chapter 3, unpredicted Netflow communications; in Chapter 6, traffic over an unusual number of destination ports. While such an approach enables known attack behaviours to be targeted, its success depends on the ability of the multi-view data to predict or explain the chosen quantity of interest. Further, unknown attack patterns will fail to be identified. In order to capture less *prescriptive* notions of anomalous behaviour, we must appeal to unsupervised learning techniques.

In this chapter, we propose a multi-view extension of multidimensional scaling (MDS), a well-known unsupervised technique for learning a latent vector representation of a dataset by preserving its pairwise dissimilarity structure. When multiple pairwise dissimilarity measurements are available, based on different views of the data, the challenge lies in dealing with disagreement;

i.e. objects that are similar in one view but dissimilar in another. We argue that the level of dependence between views lies somewhere on a continuum: at one extreme, multiple dissimilarity scores represent noisy measurements of the same pairwise relationship between objects; on the other extreme, the set of dissimilarity scores are independent, each measuring the level of dissimilarity between objects based on disjoint subsets of the true latent feature set. Our method assumes that multiple views of data are generated from *potentially intersecting subsets* of the features of the same true latent representation.

In doing so, our method embeds multi-view data into a common latent space that captures both shared and view-specific factors of variation. By considering the joint distribution of the data, as opposed to each marginal distribution, this representation has the potential to provide enhanced anomaly detection capabilities. For example, we may be able to identify the unusual *co-occurence* of observations across data sources, each of which may not be unusual within any individual data source or representation.

This idea is integral to *situational awareness* in cyber security: the continuous monitoring of assets within the IT infrastructure and the detection of any changes in behaviour over time (see e.g. Endsley (1995); Jajodia et al. (2009)). One challenge of situational awareness is deciding what quantities to monitor. By monitoring for changes taking place in a latent space that captures multiple different aspects of behaviour, we may be able to spot previously unconsidered indicators of attack. There are many ways in which the learned multi-view representation could be used in this context. In this chapter, we illustrate first steps towards a situational awareness tool based on monitoring for changes in the pairwise distance between computers in the latent space.

The remainder of the chapter proceeds as follows: In Section 7.1, we provide background to representation learning and discuss related literature. We describe single-view MDS in Section 7.2 and outline an efficient optimisation procedure. This forms the basis for our proposed multi-view extension in Section 7.3. We motivate the modified objective function and derive the optimisation scheme. In Section 7.4, the intuition behind the algorithm is

illustrated through an experiment with a toy dataset. Subsequently, using the LANL dataset, we present an initial demonstration of how our multi-view MDS algorithm could be used for situational awareness and change detection, with highly promising results.

7.1 Background and Relevant Literature

Representation learning refers to a set of techniques for finding a fixed-length vector embedding of data that makes it easier to extract useful information during downstream learning tasks. The goal is to map observed data to points in some latent space such that the locations of, or distances between, points in this space encode some meaningful property of the data.

The history of representation learning goes back to early work on principal component analysis (PCA) (Pearson, 1901) and its supervised variant, linear discriminant analysis (Fisher, 1936). A wide variety of extensions and alternatives have since been proposed to address different characteristics of the learning task. For example, kernel PCA (Schölkopf et al., 1998) is designed for datasets with non-linear structure, locally linear embedding (Roweis and Saul, 2000) attempts to preserve the local geometry of the data, while Isomap (Tenenbaum et al., 2000) attempts to preserve geometry at all scales. Most recently, a wide variety of representation learning techniques have been developed based on deep learning; for example, sparse coding (Lee et al., 2006; Olshausen and Field, 1996), autoencoders (Kramer, 1991; Vincent et al., 2008), convolutional neural networks (LeCun et al., 1998), restricted Boltzmann machines (Hinton and Salakhutdinov, 2006) and deep Boltzmann machines (Salakhutdinov and Hinton, 2009). For a review of classical and modern representation learning techniques, see Zhong et al. (2016); for emphasis on deep learning approaches, see Bengio et al. (2013). Recently, extensions to many of these methods have been proposed for the multi-view setting—see Li et al. (2019) for a review. One approach with an interesting connection to the method we develop in this chapter is CLIP (Contrastive Language-Image Pre-training) (Radford et al., 2021). CLIP separately embeds two views of data into a shared latent space so that the per-view embeddings

of each observation in this space are close together. In the method we develop, the multiple views of data are embedded into separate (but potentially overlapping) latent spaces, which each correspond to some lower-dimensional subspace of a shared latent space.

Our approach in this chapter is based on multidimensional scaling. MDS refers to a family of algorithms for modelling dissimilarity data as distances among points in a Cartesian space. Given $\boldsymbol{\Delta} \in \mathbb{R}^{n \times n}$, a symmetric, non-negative matrix of dissimilarities δ_{ij} measured between a set of n objects, MDS attempts to find $\boldsymbol{Z} \in \mathbb{R}^{n \times p}$, an embedding[1] of each object in a p-dimensional Euclidean space, such that the inter-point distances approximate the given dissimilarities. Algorithms in this family can broadly be categorised into *metric* vs. *non-metric* and *strain-* vs. *stress*-based models. In metric MDS, the objective is to achieve distances in the embedding space as close as possible to the given dissimilarities, while in non-metric MDS, only the order relations between dissimilarities are important. Strain-based, or classical, MDS (Torgerson, 1958) is an algebraic problem closely related to PCA, which can be solved by eigenvalue decomposition. Stress-based MDS (Kruskal, 1964a,b; Shepard, 1962) uses a geometric loss function which results in a non-linear and non-convex optimisation problem.

MDS has been most widely used for dimensionality reduction and visualisation applications, due to its ability to produce an intelligible 2- or 3-dimensional graphical representation of complicated high-dimensional data structures.[2] However, as we will show, the utility of MDS algorithms goes beyond data visualisation. Symbolic data, which may have extremely high-dimensional or even variable length vector representations can be naturally embedded into a latent space in which similar observations are positioned close together.

In multi-view problems, the dissimilarity of observations can be measured based on each different view; i.e. we have v different dissimilarity matrices

[1] Note we use $\boldsymbol{Z}$ to denote the embedding, in order to avoid confusion with the raw data $\boldsymbol{X}$

[2] For this reason, many graph drawing algorithms are based on MDS; e.g. Gansner et al. (2005); Kamada et al. (1989).

$\Delta^{(s)},\ s = 1, \ldots, v$. We could assume that the various dissimilarity matrices correspond to different noisy measurements of the same pairwise relationship. In this case, a multi-view MDS algorithm would attempt to find an embedding that minimises the average disagreement with respect to the dissimilarity measures. Such an approach is used by Izquierdo (2017), who proposes a multi-view version of classical MDS based on common principal components analysis (Flury, 1984; Trendafilov, 2010), and by Bai et al. (2017), who modify the objective function in MDS to minimise a weighted combination of stress functions applied to each dissimilarity matrix.

It is perhaps more natural to assume that dissimilarity matrices based on different views of data will represent different relationships between observations. Instead of trying to *compromise* between disagreeing dissimilarity measures, our latent embedding should aim to *capture* this interesting structural information. Ma et al. (2010) propose carrying out MDS on each view independently, resulting in $\boldsymbol{Z}^{(s)} \in \mathbb{R}^{n \times p_s},\ s = 1, \ldots, v$, and then concatenating these together to form the latent embedding $\boldsymbol{Z}^* \in \mathbb{R}^{n \times q}$, with $q = p_1 + \cdots + p_v$. The problem with this approach is that the solution of MDS is invariant to rotation, so multiple runs of the same algorithm will produce different embeddings, which will fail to capture the correlations between views. A better approach is that of Hossain et al. (2020), who propose a method for constructing 3d visualisations given multiple 2d projections. In what follows, we take the same approach to develop a general-purpose multi-view representation learning algorithm.

7.2 Multidimensional Scaling

In this section, we define the objective function for stress-based MDS in the single-view setting and describe an efficient optimisation procedure. Recall that MDS seeks $\boldsymbol{Z} \in \mathbb{R}^{n \times p}$ such that the inter-point Euclidean distances approximate the dissimilarities δ_{ij} stored in the matrix $\boldsymbol{\Delta} \in \mathbb{R}^{n \times n}$. The error criterion, known as stress, is given by:

$$\sigma(\boldsymbol{Z}\ ;\ \boldsymbol{\Delta}) = \sum_{i>j} \left(d_{ij}(\boldsymbol{Z}) - \delta_{ij}\right)^2, \tag{7.1}$$

where $d_{ij}(\boldsymbol{Z}) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2$. For a chosen dimensionality p, the optimum set of points $\boldsymbol{Z}^*$, known as a *configuration*, is determined by minimising $\sigma(\boldsymbol{Z} \; ; \; \boldsymbol{\Delta})$ with respect to $\boldsymbol{Z}$. In practice, this is usually achieved by setting a random initial configuration $\boldsymbol{Z}_0$ and performing a gradient-descent-type procedure until a minimum is found. We now describe one such optimisation procedure, which is guaranteed to converge monotonically to a (possibly local) minimum.

7.2.1 The SMACOF Algorithm for MDS

The expression (7.1) can be decomposed as follows:

$$\begin{aligned}\sigma(\boldsymbol{Z} \; ; \; \boldsymbol{\Delta}) &= \sum_{i>j} d_{ij}^2(\boldsymbol{Z}) - 2\delta_{ij} d_{ij}(\boldsymbol{Z}) + \delta_{ij}^2 \\ &= \operatorname{tr}\left(\boldsymbol{Z}^\top \boldsymbol{V} \boldsymbol{Z}\right) - 2\operatorname{tr}\left(\boldsymbol{Z}^\top \boldsymbol{B}(\boldsymbol{Z}) \boldsymbol{Z}\right) + \sum_{i>j} \delta_{ij}^2, \qquad (7.2)\end{aligned}$$

for matrices $\boldsymbol{V}, \; \boldsymbol{B}(\boldsymbol{Z}) \in \mathbb{R}^{n \times n}$ with elements:

$$v_{ij} = \begin{cases} -1 & i \neq j \\ n-1 & i = j, \end{cases}$$

$$b_{ij} = \begin{cases} -\frac{\delta_{ij}}{d_{ij}(\boldsymbol{Z})} & i \neq j, \; d_{ij}(\boldsymbol{Z}) \neq 0 \\ 0 & i \neq j, \; d_{ij}(\boldsymbol{Z}) = 0 \\ -\sum_{k \neq i} b_{ik} & i = j. \end{cases}$$

The objective function (7.2) is non-linear in $\boldsymbol{Z}$ and non-convex, which suggests the following first-order, gradient descent algorithm:

$$\begin{aligned}\boldsymbol{Z}_{t+1} &= \boldsymbol{Z}_t - \alpha_t \nabla \sigma(\boldsymbol{Z}_t \; ; \; \boldsymbol{\Delta}) \\ &= \boldsymbol{Z}_t - 2\alpha_t \left(\boldsymbol{V} \boldsymbol{Z}_t - \boldsymbol{B}(\boldsymbol{Z}_t) \boldsymbol{Z}_t\right).\end{aligned}$$

The step size α_t must be computed at each iteration by means of line search, which may be prohibitively expensive for large-scale problems.

De Leeuw and Heiser (1977) suggest replacing the function $\sigma(\boldsymbol{Z} \ ; \ \boldsymbol{\Delta})$ by a simpler *majorising function* $\tau(\boldsymbol{Z}, \boldsymbol{Y})$. A function $\tau(\boldsymbol{Z}, \boldsymbol{Y})$ is a majorising function if $\tau(\boldsymbol{Z}, \boldsymbol{Y}) \geq \sigma(\boldsymbol{Z} \ ; \ \boldsymbol{\Delta})$ for all $\boldsymbol{Y} \in \mathbb{R}^{n \times p}$, with equality when $\boldsymbol{Y} = \boldsymbol{Z}$. They show that by the Cauchy-Swartz inequality, the following function meets this criteria:

$$\sigma(\boldsymbol{Z} \ ; \ \boldsymbol{\Delta}) \leq \text{tr}(\boldsymbol{Z}^\top \boldsymbol{V} \boldsymbol{Z}) - 2\text{tr}(\boldsymbol{Z}^\top \boldsymbol{B}(\boldsymbol{Y})\boldsymbol{Y}) + \sum_{i>j} \delta_{ij}^2 = \tau(\boldsymbol{Z}, \boldsymbol{Y}).$$

The function $\tau(\boldsymbol{Z}, \boldsymbol{Y})$ is quadratic in $\boldsymbol{Z}$; finding its minimum involves solving

$$\begin{aligned} \nabla\tau(\boldsymbol{Z}, \boldsymbol{Y}) &= 2\boldsymbol{V}\boldsymbol{Z} - 2\boldsymbol{B}(\boldsymbol{Y})\boldsymbol{Y} = \boldsymbol{0} \\ \implies \arg\min_{\boldsymbol{Z}} \tau(\boldsymbol{Z}, \boldsymbol{Y}) &= \boldsymbol{V}^\dagger \boldsymbol{B}(\boldsymbol{Y})\boldsymbol{Y}, \end{aligned}$$

where $\boldsymbol{V}^\dagger$ is a Moore-Penrose inverse given by $\boldsymbol{V}^\dagger = n^{-1}(\boldsymbol{I}_n - n^{-1}\boldsymbol{1}_n)$.

The SMACOF algorithm (Scaling by Majorising a Complex Function) begins by constructing a random initial configuration $\boldsymbol{Z}_0$ and then iteratively minimising the majorising function $\tau(\boldsymbol{Z}, \boldsymbol{Z}_t)$, i.e. at iteration t, we find $\boldsymbol{Z}_{t+1}$ as follows:

$$\boldsymbol{Z}_{t+1} = \boldsymbol{V}^\dagger \boldsymbol{B}(\boldsymbol{Z}_t)\boldsymbol{Z}_t.$$

Gradient descent proceeds until $\sigma(\boldsymbol{Z}_t) - \sigma(\boldsymbol{Z}_{t-1}) < \epsilon$ or a certain iteration limit is reached. Pseudo-code for the algorithm is shown in Algorithm 3.

7.3 Multi-view Embedding

As discussed in Section 7.1, MDS is often pigeon-holed as purely a data visualisation tool; that is, given some dataset with many features, plot a 2-dimensional embedding of this data for the purpose of identifying structure or patterns. Despite this, it is clear from our description above that the MDS procedure can be used to discover a general p-dimensional latent Euclidean representation of a dataset, given only a dissimilarity metric between observations. Indeed, by choosing $p \gg 2$, MDS is given more degrees of freedom to better approximate the structure of the input data. Finding such a la-

Algorithm 3: SMACOF

Input:

$\boldsymbol{\Delta} \in \mathbb{R}^{n \times n}$: dissimilarity matrix

$p \in \mathbb{N}^{+}$: target dimension

$\epsilon \in \mathbb{R}$: convergence threshold

`max_iter`: maximum number of iterations

Initialise:

$\boldsymbol{Z}_0 \in \mathbb{R}^{n \times p}$

Procedure:

$\boldsymbol{V}^{\dagger} \leftarrow n^{-1}(\boldsymbol{I} - n^{-1}\mathbf{1})$

for $t = 1$ *to* `max_iter` **do**

  $\boldsymbol{Z}_t \leftarrow \boldsymbol{V}^{\dagger}\boldsymbol{B}(\boldsymbol{Z}_{t-1})\boldsymbol{Z}_{t-1}$

  if $\sigma(\boldsymbol{Z}_t \ ; \ \boldsymbol{\Delta}) - \sigma(\boldsymbol{Z}_{t-1} \ ; \ \boldsymbol{\Delta}) < \epsilon$ **then**

    break

Output:

Configuration $\boldsymbol{Z}^{*}$

tent embedding may be particularly useful if the input data is not naturally represented in vector form (e.g. symbolic data such as graphs, trees, strings) but a meaningful dissimilarity metric exists between observations. Similarly, there may be vector-valued data of variable length, e.g. DNA sequences, documents, etc. In these cases, MDS can be used to find a latent feature representation, which can serve as the input to any unsupervised learning procedure.

We are concerned with the situation when there exists multiple, disparate feature representations for each observation, which—as in our examples above—are difficult to concatenate into a vector. Our goal is to find a single, shared embedding, which jointly preserves the distance/dissimilarity structure of multiple views of the data.

7.3.1 Multi-view MDS

In the multi-view setting, we have multiple dissimilarity matrices $\boldsymbol{\Delta}^{(s)}, \ s = 1, \ldots, v$ and we are seeking a common embedding in a latent p-dimensional Euclidean space. A naïve approach would be to define the following multi-

view stress function:

$$\sigma\left(\boldsymbol{Z}\ ;\ \left\{\boldsymbol{\Delta}^{(s)}\right\}\right) = \sum_{s=1}^{v}\sum_{i>j}\left(d_{ij}\left(\boldsymbol{Z}\right) - \delta_{ij}^{(s)}\right)^2.$$

Where there is disagreement between dissimilarity measures (i.e. two points are considered similar according to one view but dissimilar according to another), this approach would find the best compromise. This would be appropriate if we assume that the various dissimilarity matrices correspond to different noisy measurements of the same pairwise relationships. However, with multi-view data, we expect that different modalities will capture different relationships between observations.

We could instead allow certain *subspaces* of the latent feature space to correspond to different views of the data. Consider as a motivating example viewing (literally) points in 3-d space. Viewed from a certain perspective, two points may appear close together even though from another angle, they are clearly far apart. The different *views* in this example correspond to different orthogonal projections of the 3-dimensional data onto 2-dimensional planes. This motivates defining a new stress function:

$$\sigma\left(\boldsymbol{Z}, \left\{\boldsymbol{P}^{(s)}\right\}\ ;\ \left\{\boldsymbol{\Delta}^{(s)}\right\}\right) = \sum_{s=1}^{v}\sum_{i>j}\left(d_{ij}\left(\boldsymbol{Z}\boldsymbol{P}^{(s)}\right) - \delta_{ij}^{(s)}\right)^2, \qquad (7.3)$$

where for each view s, we minimise the disagreement between its given dissimilarity measurements $\delta_{ij}^{(s)}$ and $d_{ij}\left(\boldsymbol{Z}\boldsymbol{P}^{(s)}\right)$, the Euclidean distances between points in the configuration, *after projection onto some lower-dimensional subspace*. Here, $\boldsymbol{P}^{(s)}$ is a $p \times p$ orthogonal projection matrix of rank q_s; that is, $\boldsymbol{P}^{(s)}$ transforms the configuration into some subspace of dimension $q_s \leq p$. To learn the optimum embedding, we must now simultaneously learn the configuration $\boldsymbol{Z}$ and the set of orthogonal projections $\left\{\boldsymbol{P}^{(s)}\right\}$.

7.3.2 Extending SMACOF for Multi-view MDS

Note that a matrix $\boldsymbol{P} \in \mathbb{R}^{p \times p}$ is a rank-q orthogonal projection if and only if $\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^\top$ for some matrix $\boldsymbol{Q} \in \mathbb{O}^{p \times q}$, where $\mathbb{O}^{p \times q}$ denotes the space of semi-orthogonal $p \times q$ matrices. As $d_{ij}(\boldsymbol{Z}\boldsymbol{P}) = d_{ij}(\boldsymbol{Z}\boldsymbol{Q})$, expression (7.3) can be decomposed as follows:

$$\begin{aligned}\sigma\left(\boldsymbol{Z}, \{\boldsymbol{Q}^{(s)}\}\ ;\ \{\boldsymbol{\Delta}^{(s)}\}\right) &= \sum_{s=1}^{v}\sum_{i>j} d_{ij}^2(\boldsymbol{Z}\boldsymbol{Q}^{(s)}) - 2\delta_{ij}^{(s)} d_{ij}(\boldsymbol{Z}\boldsymbol{Q}^{(s)}) + (\delta_{ij}^{(s)})^2 \\ &= \sum_{s=1}^{v} \operatorname{tr}(\boldsymbol{Z}^\top \boldsymbol{V}\boldsymbol{Z}\boldsymbol{Q}^{(s)}\boldsymbol{Q}^{(s)\top}) \\ &\quad - 2\operatorname{tr}(\boldsymbol{Z}^\top \boldsymbol{B}(\boldsymbol{Z}\boldsymbol{Q}^{(s)})\boldsymbol{Z}\boldsymbol{Q}^{(s)}\boldsymbol{Q}^{(s)\top}) + C,\end{aligned}$$

where C is a constant term. This is differentiable in terms of both $\boldsymbol{Z}$ and each $\boldsymbol{Q}^{(s)}$. We can therefore perform coordinate descent (Friedman et al., 2010); within each iteration we cycle through $\boldsymbol{Z}, \boldsymbol{Q}^{(1)}, \ldots, \boldsymbol{Q}^{(v)}$, updating one component while keeping the others fixed. To do so, we can use the same approach as SMACOF (Section 7.2.1) by defining the following majorising functions:

To majorise $\boldsymbol{Z}$:

$$\begin{aligned}\tau\left(\boldsymbol{Z}, \boldsymbol{Y}\right) &= \operatorname{tr}\left(\boldsymbol{Z}^\top \boldsymbol{V}\boldsymbol{Z}\sum_{s=1}^{v}\boldsymbol{Q}^{(s)}\boldsymbol{Q}^{(s)\top}\right) \\ &\quad - 2\operatorname{tr}\left(\boldsymbol{Z}^\top \sum_{s=1}^{v}\boldsymbol{B}(\boldsymbol{Y}\boldsymbol{Q}^{(s)})\boldsymbol{Y}\boldsymbol{Q}^{(s)}\boldsymbol{Q}^{(s)\top}\right) + C \\ &\geq \sigma\left(\boldsymbol{Z}\ ;\ \{\boldsymbol{Q}^{(s)}\}, \{\boldsymbol{\Delta}^{(s)}\}\right);\end{aligned}$$

To majorise $\boldsymbol{Q}^{(s)}$:

$$\begin{aligned}\tau\left(\boldsymbol{Q}^{(s)}, \boldsymbol{D}\right) &= \operatorname{tr}\left(\boldsymbol{Q}^{(s)\top}\boldsymbol{Z}^\top \boldsymbol{V}\boldsymbol{Z}\boldsymbol{Q}^{(s)}\right) - 2\operatorname{tr}\left(\boldsymbol{Q}^{(s)\top}\boldsymbol{Z}^\top \boldsymbol{B}(\boldsymbol{Z}\boldsymbol{D})\boldsymbol{Y}\boldsymbol{D}\right) + C \\ &\geq \sigma\left(\boldsymbol{Q}^{(s)}\ ;\ \boldsymbol{Z}, \{\boldsymbol{Q}^{(t)}\}_{t\neq s}, \{\boldsymbol{\Delta}^{(s)}\}\right).\end{aligned}$$

This leads to the following gradient descent equations:

$$\begin{aligned}\boldsymbol{Z}_{t+1} &= \boldsymbol{V}^{\dagger}\left(\sum_{s=1}^{v}\boldsymbol{B}(\boldsymbol{Z}_t\boldsymbol{Q}^{(s)})\boldsymbol{Z}_t\boldsymbol{Q}^{(s)}\boldsymbol{Q}^{(s)\top}\right)\left(\sum_{s=1}^{v}\boldsymbol{Q}^{(s)}\boldsymbol{Q}^{(s)\top}\right)^{-1};\\ \widetilde{\boldsymbol{Q}}_{t+1}^{(s)} &= \left(\boldsymbol{Z}^{\top}\boldsymbol{V}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^{\top}\boldsymbol{B}(\boldsymbol{Z}\boldsymbol{Q}_t^{(s)})\boldsymbol{Z}\boldsymbol{Q}_t^{(s)}.\end{aligned} \tag{7.4}$$

Note that equation (7.4) is not sufficient for updating $\boldsymbol{Q}^{(s)}$. This is because $\mathbb{O}^{p\times q}$ is not a subspace of $\mathbb{R}^{p\times q}$. To overcome this, we need to follow our gradient step with a projection back into the required space. The projection of $\widetilde{\boldsymbol{Q}} \in \mathbb{R}^{p\times q}$ onto $\mathbb{O}^{p\times q}$ is given by

$$\underset{\boldsymbol{Q}\in\mathbb{O}^{p\times q}}{\arg\min}\ \|\widetilde{\boldsymbol{Q}} - \boldsymbol{Q}\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm. $\boldsymbol{Q}$ can easily be computed via singular value decomposition of $\widetilde{\boldsymbol{Q}}$: if $\widetilde{\boldsymbol{Q}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}$, then $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^{\top}$. Pseudocode for multi-view SMACOF is shown in Algorithm 4.

Note that while SMACOF algorithms guarantee a series of non-increasing stress values, they may converge to a local minimum. However, as De Leeuw and Mair (2009) point out, local minima are more likely to occur in low-dimensional solutions and are rather unlikely in high-dimensional solutions. As we are using MDS for representation learning, as opposed to data visualisation, we are not required to seek a low-dimensional solution and so do not expect local minima to present a problem.

A further important point to note is that we have so far assumed knowledge of both the desired embedding dimensionality p and the ranks of the per-view projections $q_1, \ldots, q_v$. In reality, the optimum choice will need to be estimated from the data. One option is to search over the set of possible combinations of embedding dimensionality and projection ranks to find the model that results in the lowest stress. However, for a fixed embedding dimensionality, an exhaustive search over projection ranks would require testing p^v combinations, which may be prohibitive. In Section 7.4.2, we will describe a practical approach to solving this problem.

Algorithm 4: Multi-view SMACOF

Input:

$\boldsymbol{\Delta}^{(1)}, \ldots, \boldsymbol{\Delta}^{(v)} \in \mathbb{R}^{n \times n}$: dissimilarity matrices

$p \in \mathbb{N}^{+}$: target dimension

$q_1, \ldots, q_v \in \mathbb{N}^{+},\ q_s \leq p$: subspace dimension for each view

$\epsilon \in \mathbb{R}$: convergence threshold

`max_iter`: maximum number of iterations

Initialise:

$\boldsymbol{Z}_0 \in \mathbb{R}^{n \times p}$

$\boldsymbol{Q}_0^{(1)} \in \mathbb{O}^{p \times q_1}, \ldots, \boldsymbol{Q}_0^{(v)} \in \mathbb{O}^{p \times q_v}$

Procedure:

$\boldsymbol{V}^{\dagger} \leftarrow n^{-1}(\boldsymbol{I} - n^{-1}\boldsymbol{1})$

for $t = 1$ *to* `max_iter` **do**

$\boldsymbol{Z}_t \leftarrow$
$\boldsymbol{V}^{\dagger} \left(\sum_{s=1}^{v} \boldsymbol{B}(\boldsymbol{Z}_{t-1}\boldsymbol{Q}_{t-1}^{(s)})\boldsymbol{Z}_{t-1}\boldsymbol{Q}_{t-1}^{(s)}\boldsymbol{Q}_{t-1}^{(s)\top} \right) \left(\sum_{s=1}^{v} \boldsymbol{Q}_{t-1}^{(s)}\boldsymbol{Q}_{t-1}^{(s)\top} \right)^{-1}$

for $s = 1$ *to* v **do**

$\widetilde{\boldsymbol{Q}}_t^{(s)} = \left(\boldsymbol{Z}_t^{\top}\boldsymbol{V}\boldsymbol{Z}_t\right)^{-1} \boldsymbol{Z}_t^{\top}\boldsymbol{B}(\boldsymbol{Z}_t\boldsymbol{Q}_{t-1}^{(s)})\boldsymbol{Z}_t\boldsymbol{Q}_{t-1}^{(s)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}$

$\boldsymbol{Q}_t^{(s)} \leftarrow \boldsymbol{U}\boldsymbol{V}^{\top}$

if $\sigma(\boldsymbol{Z}_t, \{\boldsymbol{Q}_t^{(s)}\}\ ;\ \{\boldsymbol{\Delta}^{(s)}\}) - \sigma(\boldsymbol{Z}_{t-1}, \{\boldsymbol{Q}_{t-1}^{(s)}\}\ ;\ \{\boldsymbol{\Delta}^{(s)}\}) < \epsilon$ **then**

break

Output:

Configuration $\boldsymbol{Z}^{*}$

7.4 Experiments

In this section, we present experimental results using the multi-view embedding procedure just presented. First, using synthetically generated data, we demonstrate the algorithm's ability to embed multi-view data into a shared latent space that accounts for correlation between views. Subsequently, we present an experiment designed to illustrate how the method could be employed for cyber security situational awareness.

7.4.1 Simulated Data

The multi-view MDS method is designed to embed separate views of a dataset into potentially overlapping subspaces of the same shared latent space. The

following experiment, using a simulated toy dataset, is designed to reinforce this concept.

Data Generation

Inspired by the technique of orthographic projection in engineering and technical drawing (see e.g. Shah and Rana (2009, ch. 12)), we simulate data comprising points within an irregular 3-d shape and construct three 'views' of this data by taking the orthogonal projections onto the xy, xz and yz planes. The data and corresponding views are depicted in Figure 7.1. Two points are shown in red to highlight disagreement between views; in the yz view, the points appear very close, despite being far apart in both the xy and xz views.

This example visually conveys the intuition behind our approach to multi-view representation learning. The assumption is that there exists some latent representation of the data, from which all views are generated. Instead of corresponding to multiple noisy measurements of the true data, multiple views each represent incomplete snapshots of the true data. Different views capture different—but not necessarily disjoint—*subsets* of the total information contained in the data's latent representation. The goal of multi-view representation learning is to reconstruct or approximate this latent representation, taking into account both the commonality and disparity between the information conveyed by different views.

Evaluation

As shown in Figure 7.1, we have three views of the data: $\boldsymbol{x}_i^{(s)} \in \mathbb{R}^2,\ s = 1, 2, 3,\ i = 1, \ldots, n$. We construct dissimilarity matrices by calculating the inter-point Euclidean distances:

$$\boldsymbol{\Delta}_{ij}^{(s)} = \|\boldsymbol{x}_i^{(s)} - \boldsymbol{x}_j^{(s)}\|^2,\ s = 1, 2, 3.$$

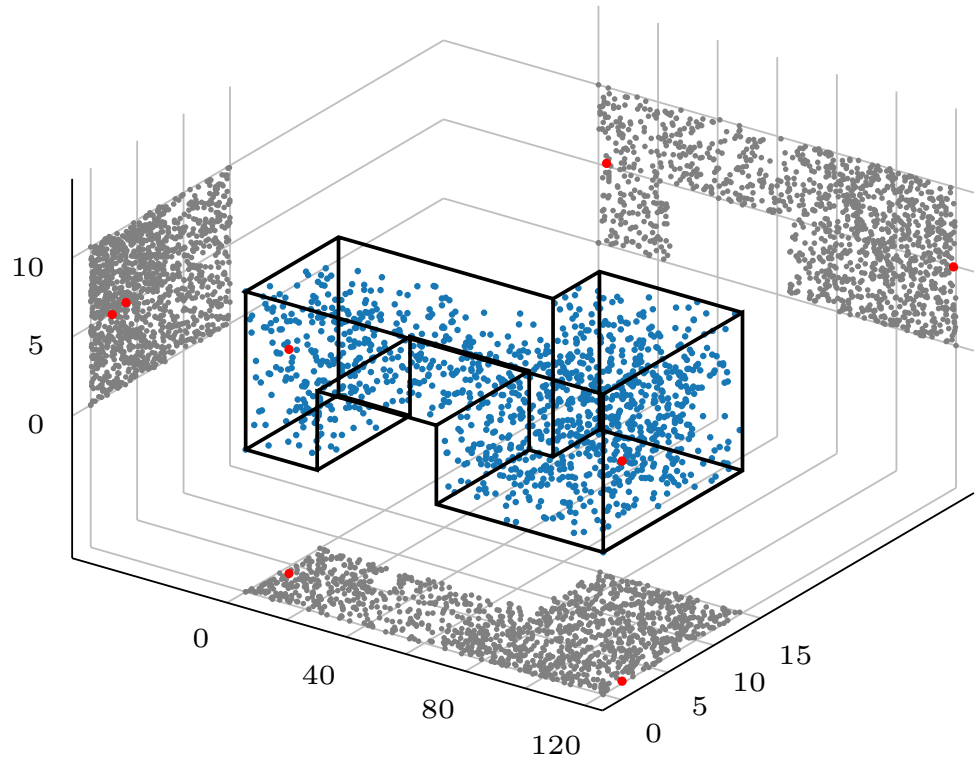
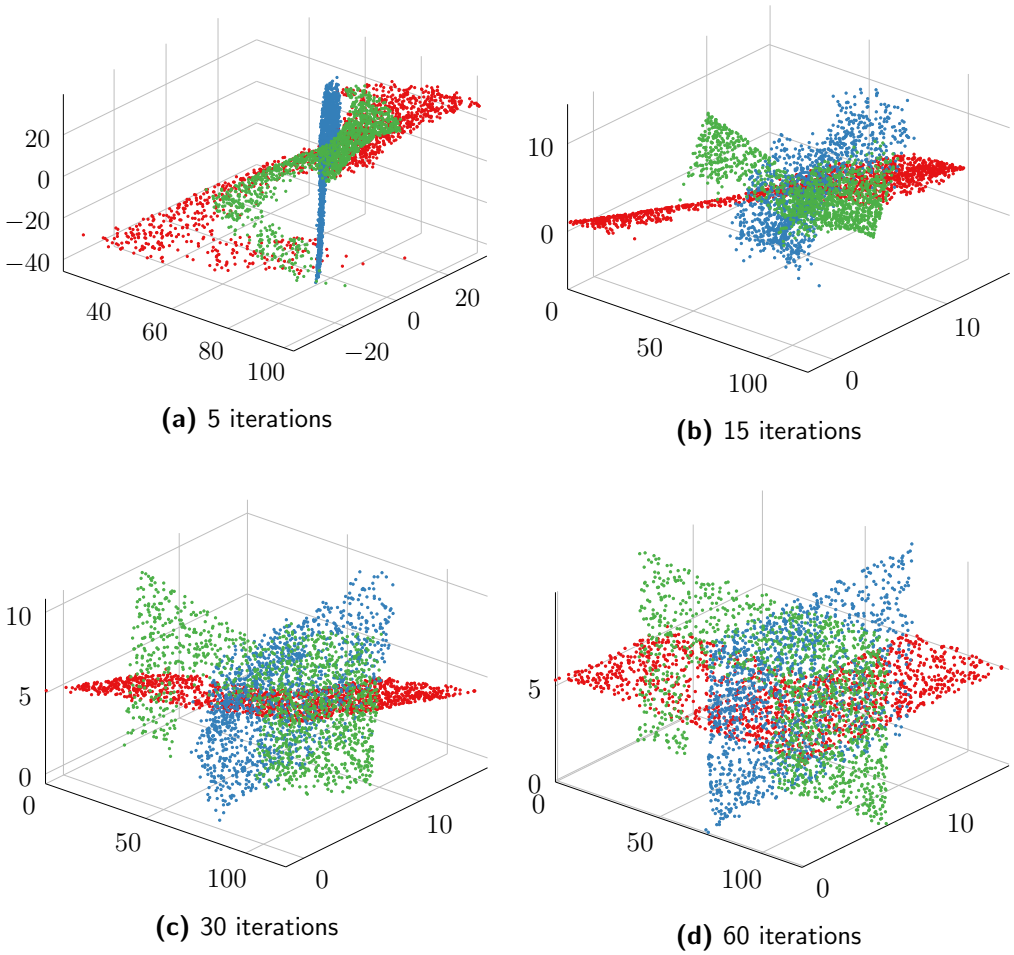


Figure 7.1: Simulated data and different views under orthogonal projections. Two example points are shown in red, which appear close together in one view but are far apart in the other two views.

The per-view data can be discarded, as the multi-view SMACOF algorithm has access only to these three sets of inter-point dissimilarities. The algorithm is initialised with a set of n randomly sampled points $\boldsymbol{Z}_0 \sim N(\boldsymbol{0}_3, \boldsymbol{I}_3)$; matrices $\widetilde{\boldsymbol{Q}}_0^{(s)} \in \mathbb{R}^{3\times 2}$ are similarly sampled before being projected onto $\boldsymbol{Q}_0^{(s)} \in \mathbb{O}^{3\times 2}$ as described in Section 7.3.2.

In each iteration, the algorithm first adjusts the positions of all points in the configuration $\boldsymbol{Z}_t$ to reduce the value of the stress criterion, given the current projection matrices. Then, given the positions of the points in the configuration, the algorithm finds—for each view in turn—a new matrix $\boldsymbol{Q}_t^{(s)}$ such that the projection of the data $\boldsymbol{Z}_t\boldsymbol{Q}_t^{(s)}$ better captures the distance structure implied by that view's dissimilarity matrix $\boldsymbol{\Delta}^{(s)}$.

In Figure 7.3, we plot the projections $\boldsymbol{Z}_t\boldsymbol{Q}_t^{(s)}\boldsymbol{Q}_t^{(s)\top}$ after various numbers of iterations. After 30 iterations, the configuration appears to be moving in the



(a) 5 iterations

(b) 15 iterations

(c) 30 iterations

(d) 60 iterations

Figure 7.2: Fitted projections after various numbers of iterations.

right direction; after 60 iterations, the algorithm has reconstructed the true configuration.

To highlight the difference between this joint configuration/projection approach and standard MDS, we compare it with the following approaches:

- MDS_{avg}: We apply single-view metric MDS (i.e. Algorithm 3), using the following distance matrix:

$$\boldsymbol{\Delta}_{\text{avg}} = \frac{1}{3}\sum_{s=1}^{3} \boldsymbol{\Delta}^{(s)}.$$

By using this method, we assume that all views represent noisy measurements of the *same* pairwise relationship. If noise in the observed dissimilarity matrices is assumed to be independent and identically distributed, then $\boldsymbol{\Delta}_{\text{avg}}$ is an unbiased estimate of the true distance matrix.

- MDS_{sum}: We apply single-view metric MDS, with

$$\boldsymbol{\Delta}_{\text{sum}} = \sqrt{\sum_{s=1}^{3} \left(\boldsymbol{\Delta}^{(s)}\right)^2}.$$

  This assumes that each view contains *disjoint subsets* of information. As MDS models dissimilarities as Euclidean distances, the total interpoint distance is given by the Euclidean sum of view-specific distances.

In Figure 7.3, we visually compare the results. The fitted configuration from an MDS algorithm is invariant to rotation, so the points are first rotated to re-align with the standard axes. As the shapes of the configurations are hard to discern in a 3d plot, we show 2d projections onto the xy and xz planes. Clearly, multi-view SMACOF is the only algorithm that is able to reconstruct the true data.

Note that MDS_{sum} fails in this example due to correlation between views. The resulting configuration is distorted due to individual components of the true interpoint distances contributing to multiple terms inside the Euclidean summation. If the different views were truly disjoint—i.e. if we had constructed dissimilarity matrices from the x, y and z components separately—then we would expect MDS_{sum} to reconstruct the true data. Multi-view SMACOF would naturally allow for this situation also by learning rank-1 projection matrices $\boldsymbol{Q}^{(s)} \in \mathbb{O}^{p\times 1}$.

7.4.2 Cyber Situational Awareness

With this experiment, we demonstrate how our multi-view representation learning algorithm could be used for cyber security situational awareness. The idea is to represent each computer's activity during a given time interval

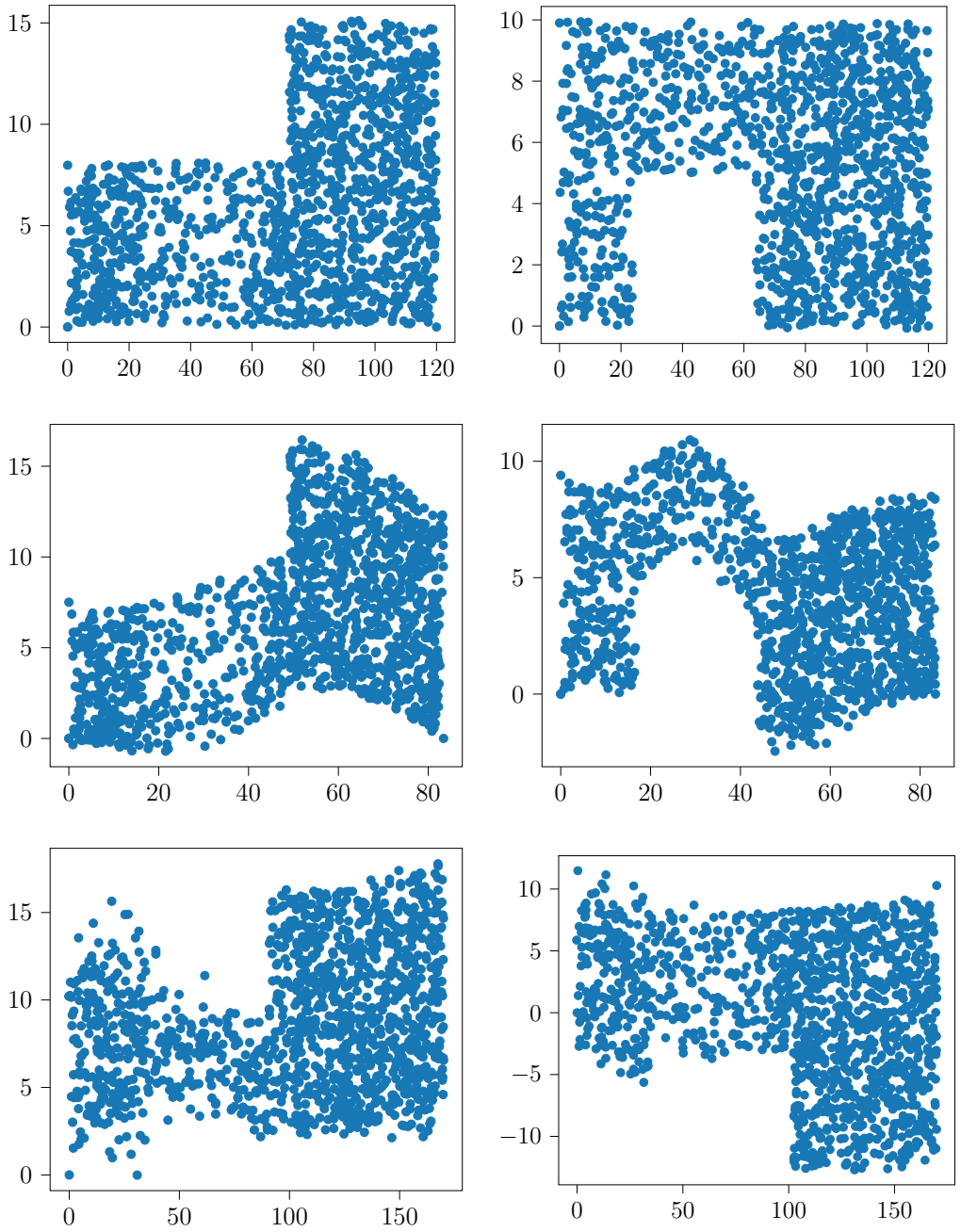


Figure 7.3: Two-dimensional views of the fitted three-dimensional configurations from multi-view SMACOF (top row), MDS_{avg} (middle row) and MDS_{sum} (bottom row). In each case, the left column shows the xy perspective and the right column shows the xz perspective.

as a point in a shared latent space and to monitor for changes that occur in this space over time.

Data and Dissimilarity Measures

As in Chapter 6, we divide the LANL data into 15-minute bins. Each computer can then be considered based on three different views of the data, which capture different aspects of its activity: *who* uses the computer, *what* processes run on the computer and *where* does network traffic flow. We define the following views and corresponding dissimilarity measures:

- **Netflow:** As described in Section 3.2, the Netflow data during a given time interval can be considered as a graph containing nodes C_i and edges $a_{i,j} \in \{0, 1\}$ indicating the presence/absence of a communication between C_i and C_j. For each bin, we construct $\boldsymbol{x}_i^{(1)} = \{C_j \ ; \ a_{i,j} = 1\}$, i.e. the set of devices with which computer i communicates. To measure dissimilarity between computers, we use the Jaccard distance (Jaccard, 1912):
$$\delta_{ij}^{(1)} = 1 - \frac{|\boldsymbol{x}_i^{(1)} \cap \boldsymbol{x}_j^{(1)}|}{|\boldsymbol{x}_i^{(1)} \cup \boldsymbol{x}_j^{(1)}|}.$$

- **Processes:** For each computer, we gather from the WLS dataset a list of all the processes that run on the device during each bin. We use a tf-idf bag-of-words representation and the cosine distance function:
$$\delta_{ij}^{(2)} = 1 - \frac{\boldsymbol{x}_i^{(2)\top} \boldsymbol{x}_j^{(2)}}{\|\boldsymbol{x}_i^{(2)}\| \|\boldsymbol{x}_j^{(2)}\|}.$$

- **Authentications:** We consider the outbound authentication activity from each computer. According to Turcotte et al. (2018), the `DomainName` field, combined with the `UserName` field should be considered a unique account identity. We therefore inspect all authentication events for which a given computer is listed as the `Source`, and gather a list of user account identifiers. Again we use the Jaccard distance to

measure dissimilarity:

$$\delta_{ij}^{(3)} = 1 - \frac{|\boldsymbol{x}_i^{(3)} \cap \boldsymbol{x}_j^{(3)}|}{|\boldsymbol{x}_i^{(3)} \cup \boldsymbol{x}_j^{(3)}|}.$$

Multi-view Embedding

As discussed in Section 7.3.2, the quality of the learned multi-view representation depends on the choice of latent dimensionality and the rank of each projection. These typically will not be known *a priori* and should be estimated from the data. In single-view MDS problems, the overall quality of an embedding is typically measured using a normalised version of the stress function, known as Kruskal's type-1 stress:

$$\sigma_1(\boldsymbol{Z} \; ; \; \boldsymbol{\Delta}) = \sqrt{\frac{\sum_{i>j} \left(d_{ij}(\boldsymbol{Z}) - \delta_{ij}\right)^2}{\sum_{i>j} \delta_{ij}^2}}. \tag{7.5}$$

The dimensionality p should be chosen sufficiently large to achieve a value of $\sigma_1(\boldsymbol{Z} \; ; \; \boldsymbol{\Delta})$ below some desired threshold. Based on guidelines by Kruskal (1964a), a commonly used threshold is 0.05. However, it is important to note that this guideline is based on empirical experience, rather than theoretical criteria. The stress of an MDS solution will depend on various factors, such as the number of points, the dimensionality of the configuration, the kind and amount of error in the input dissimilarities, etc. As such, guideline thresholds should be used with care and other diagnostic tests should be considered when selecting a threshold—see e.g. Borg and Groenen (2005, ch. 3).

For each individual view, we can therefore perform single-view MDS (Algorithm 3) to determine the dimensionality required to achieve a sufficiently high-quality fit. Figure 7.4 shows a scree plot (Cattell, 1966) of the normalised stress from MDS configurations for increasing values of p. We use this plot to inform our choice of projection ranks q_s, $s = 1, 2, 3$ in Algorithm 4. Evidently, 9 dimensions are required to achieve a high-quality re-

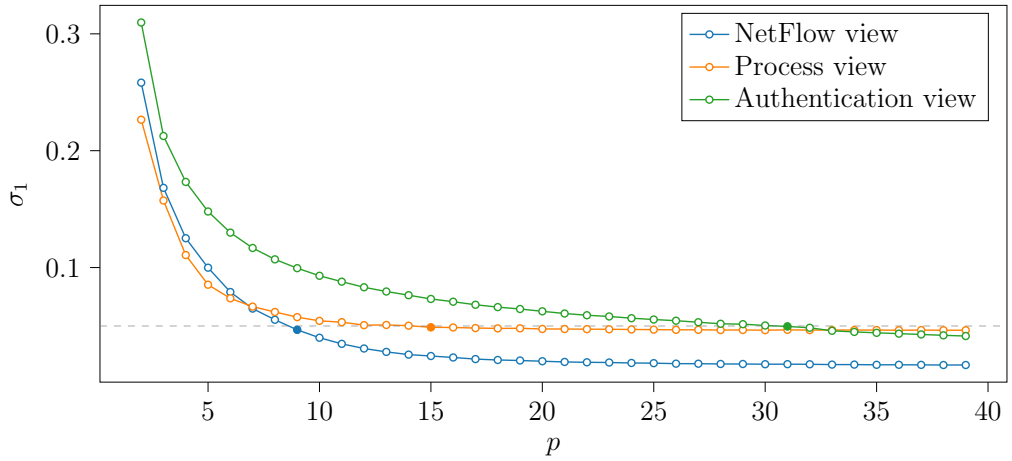


Figure 7.4: Three scree plots obtained by fitting single-view MDS models to the dissimilarity matrices of each view separately. The filled markers indicate the value of p for which a model achieved a normalised stress below the threshold 0.05.

construction of the NetFlow view; 15 and 31 dimensions are required for the process and authentication views, respectively.

If the three views of data were independent, then the shared configuration would require $p = \sum q_s = 55$ dimensions in order for the three projections of the configuration to achieve the same level of quality as their single-view MDS counterparts. As was illustrated through the toy dataset experiment, the resulting configuration would be equivalent to performing MDS_{sum}. If there is shared information among the views, then a value of $31 \leq p \leq 55$ will allow certain dimensions of the configuration to contribute to multiple projections; that is, multiple projections will have shared structure. To find the optimum latent dimensionality, we can repeat the same procedure as before, this time re-fitting multi-view SMACOF for increasing values of p until the normalised stress of the configuration under each projection is below our threshold: $\sigma_1\left(\boldsymbol{Z}\boldsymbol{P}^{(s)}\,;\,\boldsymbol{\Delta}^{(s)}\right) \leq 0.05,\ s = 1, 2, 3$. In Figure 7.5, this criterion is met for $p = 48$; that is, we can learn a representation of the multi-view data in $\mathbb{R}^{48}$, such that certain 'features' of this representation capture information shared by multiple different views.

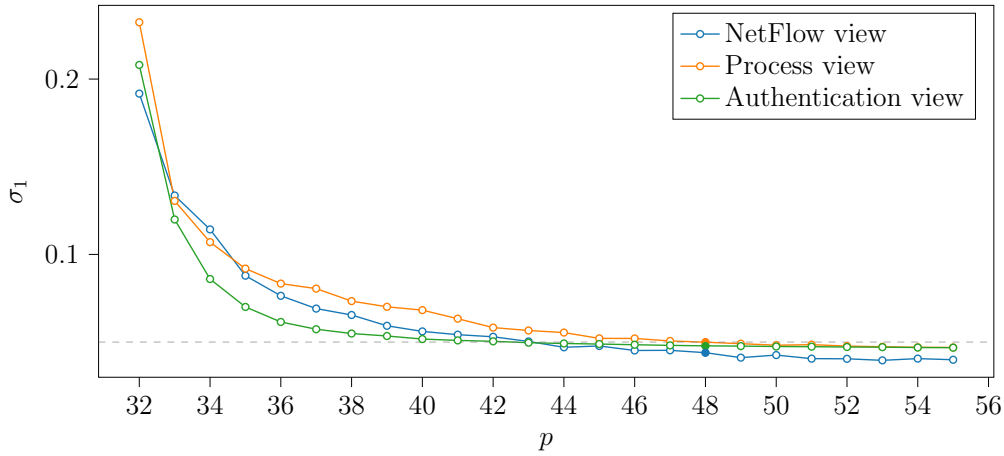


Figure 7.5: Scree plots showing the per-view normalised stress of the multi-view SMACOF configuration; i.e. $\sigma_1\left(\boldsymbol{Z}\boldsymbol{P}^{(s)}\ ;\ \boldsymbol{\Delta}^{(s)}\right)$ for $s = 1, 2, 3$. For $p = 48$, all three components of the total stress are below 0.05.

Detecting an Artificial Change in Behaviour

Each computer, during bin t of data, can be represented by $\boldsymbol{z}_{i,t} \in \mathbb{R}^{48}$. As MDS configurations are invariant to rotation, it is meaningless to monitor $\boldsymbol{z}_{i,t}$ directly as a time series for contiguous values of t. However, there are many other options available to monitor for changes within the latent space; for example clustering methods, density estimation, etc. One approach is to monitor the inter-computer distances, which are unaffected by rotation. In particular, we could monitor each computer's distance from a specific computer of known importance. One example computer within an organisation that is of paramount concern is the *Active Directory*. This computer authenticates and authorises all user accounts and computers in the network, assigning and enforcing security policies. If a malicious actor gains access to the Active Directory, they hold the keys to all the computers on the network. It may therefore be interesting to monitor a computer's distance from the Active Directory in latent space and detect any abrupt (or indeed gradual) changes over time.

To illustrate this idea, we perform multi-view MDS for each 15-minute bin of data during a day, using a sample of 1,000 computers from the LANL dataset. For each $\boldsymbol{z}_{i,t}$, we use the Mahalanobis distance (Mahalanobis, 1936) to mea-

sure how far each computer is from the Active Directory in latent space. This is shown for a random sample of 100 computers in Figure 7.6. Highlighted in black are two example computers, one of which is consistently 'closer' to the Active Directory than the other. To simulate the situation where a computer on the network abruptly changes behaviour, from bin 30 to 96, we artificially alter the raw data corresponding to the farther computer by replacing its activity with that of the closer. We refit the multi-view MDS algorithm using the resulting modified dissimilarity matrices. In the top panel of Figure 7.7, we plot the latent distance between the doctored computer and the Active Directory. A clear change can be seen. This is entirely expected, based on how we altered the data. However, it is interesting to note that when we examine the individual view-specific dissimilarity measures (bottom two panels of Figure 7.7), the change is considerably less obvious. This is due to the fact that a computer's position in latent space is determined based on its dissimilarities with respect to *all* other computers.

In practice, upon detecting a change such as that shown in Figure 7.7, a security analyst would triage to understand the cause of the change in latent space. They may discover, for example, that the set of processes being run on the device changed in some measurable way. If this was a quantity that was not previously being monitored, its discovery could help inform the design of future signature-based anomaly detection techniques. This reflects the OODA (observe-orient-decide-act) loop (Boyd, 1996), a well known framework in situational awareness.

Red Team

A distinguishing feature of the LANL dataset is the presence of so-called *red team* activity within the data. During days 57 through 82, the security team at LANL tested the robustness of the network by mimicking a cyber attack. One host, `Comp215429`, was selected as the initially compromised host, from which the red team attempted to compromise other network hosts. With this knowledge in hand, we performed multi-view embedding using the same views as previously to investigate whether any unusual activity would be

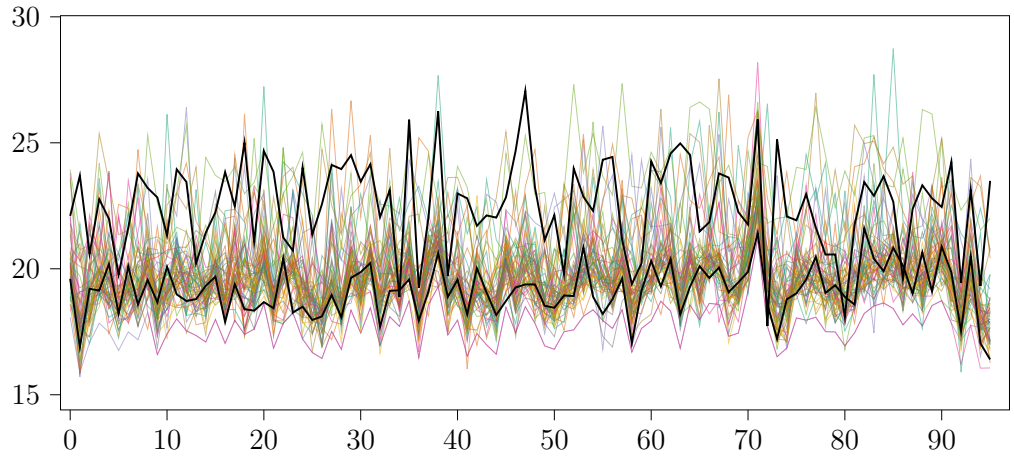


Figure 7.6: Mahalanobis distance between a sample of 100 computers and the Active Directory in latent space. Two example computers are highlighted in black with different average distances from the Active Directory in latent space.

apparent in the resulting time series of inter-computer distances.

In Figure 7.8, we show the distance between `Comp215429` and the Active Directory in latent space between days 56 and 60. A substantial spike is evident on day 59. When we examine the raw data, we learn that some NetFlow activity takes place on the computer in days 57 and 58, while the first outbound authentication event to another device occurs on day 59, corresponding with this spike. Closer inspection of the data reveals an unusual pattern. During most 15-minute bins, authentications take place from thousands of computers on the network; however, during the bin corresponding to the spike, `Comp215429` is the only computer active in the authentication view. This is clearly an artefact of the red team exercise and this specific change in network behaviour would likely be detected by numerous other situational awareness tools. That said, this example still illustrates a promising aspect of our proposed tool: the ability to detect arbitrary changes in network activity, which may or may not be monitored elsewhere.

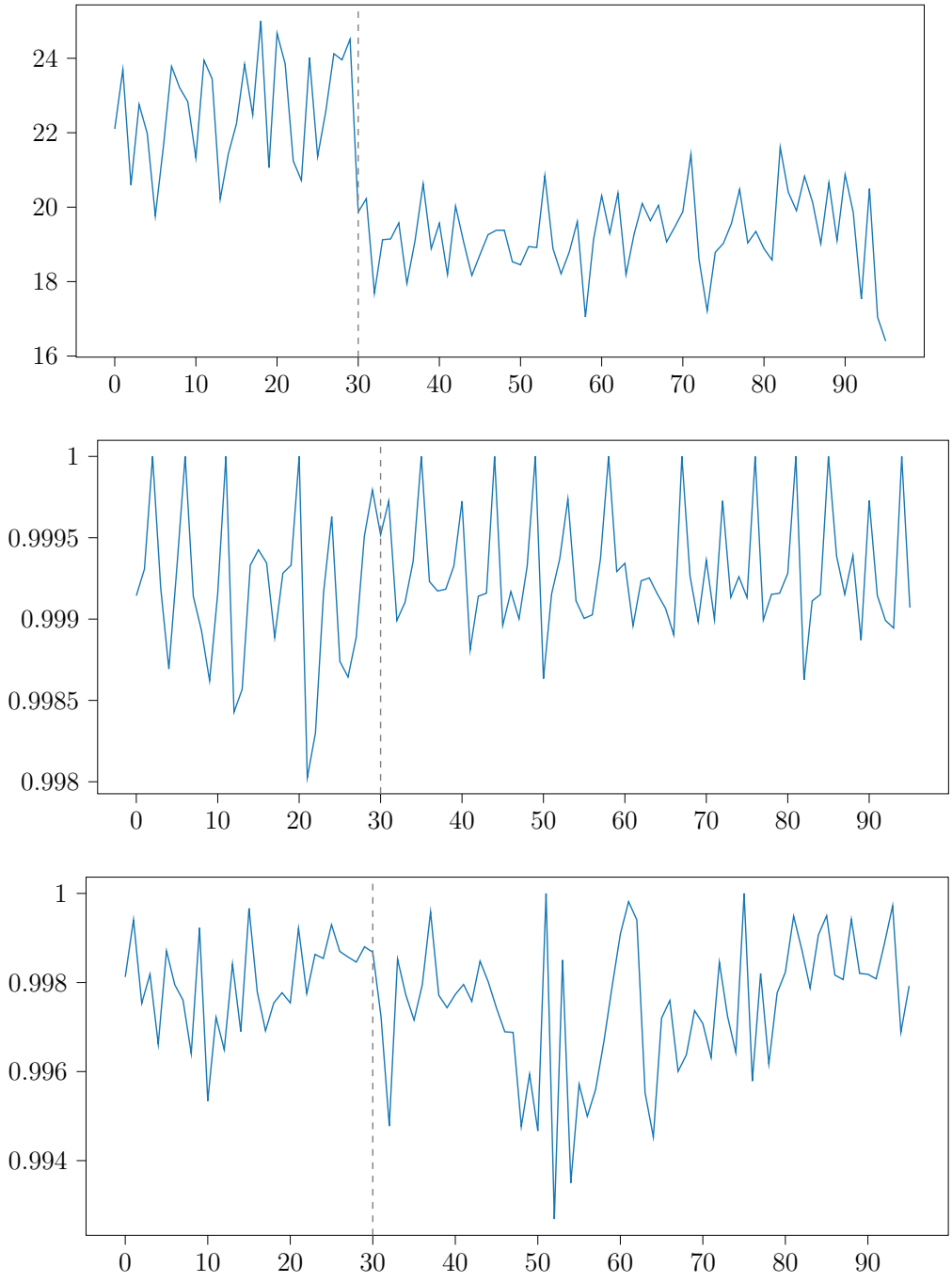


Figure 7.7: Time series of distances between the artificially doctored computer and the Active Directory. A dashed line indicates when the raw data was modified. Top: Mahalanobis distance between computers in latent space. Middle: Jaccard distance between computers in the NetFlow view. Bottom: cosine distance between computers in the Process view. Note the Authentication view is not shown, as the distance remained constant.

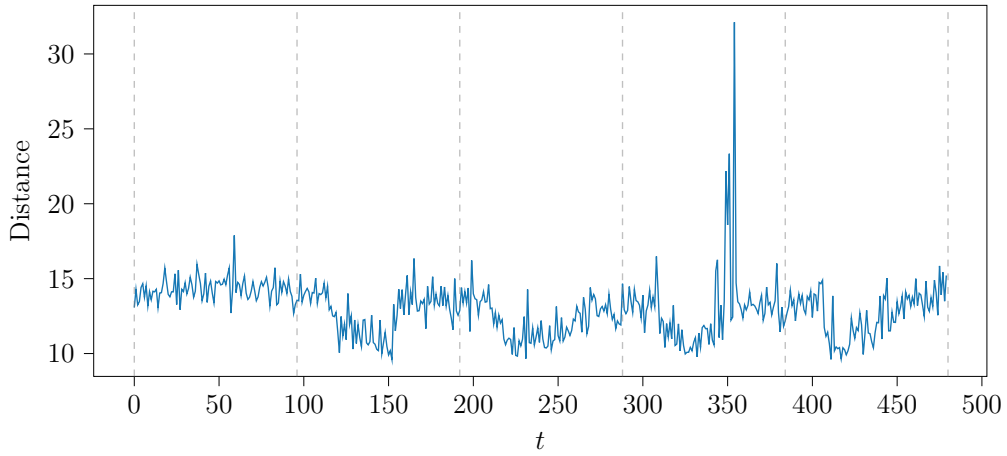


Figure 7.8: Distance between `Comp215429` and the Active Directory during the red team security exercise. The dashed lines indicate days 56 to 60.

7.5 Discussion

In this chapter, we have presented a novel framework for learning a shared latent representation of multi-view data. We extended the SMACOF algorithm to jointly optimise a shared configuration and per-view projection matrices. We presented a procedure for choosing the dimensionality of the configuration and the ranks of the projection matrices, which determine the level of shared information between views. An experiment with toy data illustrated the intuition behind the model, and highlighted how our method fills the gap between two edge cases of multi-view MDS relating to the level of dependence between views. When either of these two approaches is in fact appropriate for a given dataset, our method recovers it as a special case. Finally, we presented a simple example of how multi-view MDS could be used for situational awareness in the cyber security context. We demonstrated the detection of both an artificial and a real change in the network activity. The results suggest that multi-view MDS may be a promising tool for cyber security situational awareness, which could be further developed with the assistance of deeper domain knowledge.

8

Conclusion

The work presented in this thesis was motivated by various challenges that are presented when attempting to apply statistical approaches to cyber security. First, cyber data is multi-view: a given aspect of a network or a piece of software can be simultaneously represented by a combination of graphs, lists, vectors, histograms, trees, etc. Second, labelled data indicating benign and malicious observations is either scarce or completely unavailable. When labelled data is scarce, semi-supervised learning approaches may be effective; these were considered in Chapter 5. Otherwise, a lack of labelled data calls for anomaly detection approaches. We proposed three approaches for multi-view anomaly detection:

In Chapter 3, we proposed a framework for contriving labelled data so that classification techniques can be deployed for novelty detection. The idea behind the anomaly detection framework is to draw attention to events that are observed despite a low predicted probability of occurrence; i.e. observations with high *residuals*. In demonstrating the framework, we explored the use of naïve early and late fusion strategies for multi-view learning. Although both approaches are sub-optimal, the results highlighted that dependence exists between features from multiple views of the LANL dataset.

In Chapter 6, we proposed a strategy that characterises observations as anomalous based not only on their residuals but also their *leverage*. In order to identify such observations in multi-view data, we required a more sophisticated modelling technique. We proposed a novel multi-view supervised learning algorithm, which is capable of learning to predict an arbitrary response type using complicated input data with multiple diverse feature representations. Our model extends the classical generalised linear model to operate in the vector-valued RKHS implied by an operator-valued multi-view kernel. This kernel can be tailored to the characteristics of multiple views of the input data, which may be vector-valued or symbolic. The flexibility and strong predictive performance of the model was demonstrated on a variety of datasets, as well as its ability to detect outlier points.

The MVGKM algorithm of Chapter 6 implicitly maps multi-view observations into a vector-valued RKHS, wherein a linear model is fit. Observations with high leverage correspond to points in outlying regions of this implicit feature space. Inspired by this, in Chapter 7, we proposed a novel algorithm for *explicitly* mapping multi-view data into a shared latent space. Using a modified multidimensional scaling objective function, we proposed a SMACOF-style optimisation procedure, which embeds multi-view data into a latent space in a way that simultaneously preserves multiple notions of distance between observations. This feature representation can then be used for any unsupervised algorithm, such as clustering or outlier detection. We demonstrated one potential application of the method: cyber security situational awareness.

Finally, it should be noted that cyber security is just one of the many interesting potential application areas for the novel methods we developed in this thesis. With technological advances comes an ability to gather and store data in increasingly granular and structure-rich representations. Statistical models based on appropriately designed kernel and distance functions have a proven ability to identify signals *within* complicated data structures. We hope this thesis has contributed to the understanding that multi-view learning—including the methods we have proposed—can and should be used to uncover signals hidden both within and *between* diverse views of data.

Future Work

To conclude, we provide a brief discussion of some limitations of the methods described in this thesis and interesting opportunities for future work.

Chapter 4 presented a thorough exploration of various methods of averaging ROC curves and provided clear and reasoned guidance on how to select the appropriate method for a given study. A known limitation of ROC curves is their tendency to provide misleading or optimistic assessments of classifier performance in the presence of extreme class imbalance (Davis and Goadrich, 2006). In these situations, the Precision/Recall curve is often more meaningful. It would be interesting therefore to extend the study of averaging techniques to Precision/Recall curves.

The semi-supervised learning method developed in Chapter 5 could be extended to further leverage unlabelled data. In our formulation, a regularisation term is used to encourage agreement between the predictions of the per-view target functions for unlabelled data. An additional regularisation term could be used to restrict the hypothesis space of target functions to lie on some low-dimensional manifold. So-called *manifold regularisation* (Belkin et al., 2006) attempts to learn the geometry of the input space using unlabelled data. Minh and Sindhwani (2011) describe a regularisation term that can be used to capture possible dependencies between output variables by the use of, for example, an output graph Laplacian.

One of the main strengths of the multi-view supervised learning framework proposed in Chapter 6 was its flexibility to model a wide variety of response data types. However, the quality of the resulting model will depend on the validity of the distributional assumptions made. There is scope to improve the robustness of the framework by developing additional link functions and making any required modifications to the optimisation procedure to allow for response variables that violate the assumptions of simple exponential family distributions. For example, the Poisson likelihood function could be substituted for a *quasi-likelihood* function that can deal with over-dispersion; i.e. the case when the variability of y_i about $\widehat{\mu}_i$ is larger than what $\widehat{\mu}_i$ predicts. A different, but related, situation that often arises when modelling count

data is a seemingly "excessive" number of examples with a count of 0. A *zero-inflated* Poisson model could be developed to deal with this.

A further limitation of the MVGKM method is its computational complexity. Single-view kernel methods suffer from the requirement to invert the kernel Gram matrix, resulting in a computational cost of $\mathcal{O}(n^3)$. In the MVGKM, this is exacerbated by the presence of multiple views, leading to an $\mathcal{O}(v^3n^3)$ computational cost in most cases, and an even worse $\mathcal{O}((g-1)^3v^3n^3)$ cost for multinomial logistic regression. A popular method of reducing the computational burden of kernel methods is so-called *Nyström approximation* (Williams and Seeger, 2000), whereby the kernel Gram matrix is replaced by a low-rank approximation. Another popular method is random Fourier features (RFF) (Rahimi and Recht, 2007), which estimates low-dimensional feature embeddings such that their inner-products approximate the kernel function. An operator-valued kernel analogue of RFF has been proposed by Brault et al. (2016).

Finally, the unsupervised multi-view representation learning method developed in Chapter 7 presents an obvious avenue for future work. The motivating hypothesis was that various views of a dataset are generated from possibly overlapping subsets of the features of some shared latent representation. As such, our multi-view MDS method estimates a latent embedding so that various orthogonal projections of this embedding approximate the embeddings that would be generated from single-view MDS on the respective views. It would be interesting to apply this same construction using different embedding methods such as autoencoders. These have been proven to capture interesting non-linear structure within datasets that may not be revealed through the dissimilarity measures available to MDS.

References

N. M. Adams and D. J. Hand. Comparing classifiers when the misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.

M. A. Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

M. R. Amini, N. Usunier, and C. Goutte. Learning from multiple partially observed views — an application to multilingual text categorization. *Advances in Neural Information Processing Systems*, 22:28–36, 2009.

B. Anderson, C. Storlie, and T. Lane. Improving malware classification: Bridging the static/dynamic gap. In *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*, pages 3–14, 2012.

H. S. Anderson and P. Roth. EMBER: An open dataset for training static PE malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

R. Anthony. Detecting security incidents using Windows workstation event logs. *SANS Institute, InfoSec Reading Room Paper*, 2013.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *International Conference on Machine Learning*, pages 6–13, 2004.

S. Bai, X. Bai, L. J. Latecki, and Q. Tian. Multidimensional scaling on multiple input distance matrices. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 1281—1287, 2017.

C. R. Baker. Mutual information for Gaussian processes. *SIAM Journal on Applied Mathematics*, 19(2):451–458, 1970.

A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. In *Proceedings of the International Conference on Image Processing*, volume 3, pages III–513, 2003.

V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 1st edition, 1978.

M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe. 220 band AVIRIS hyperspectral image data set: June 12, 1992 Indian pine test site 3, 2015.

S. Becker. Mutual information maximization: Models of cortical self-organization. *Network: Computation in Neural Systems*, 7(1):7–31, 1996.

S. Becker and G. E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(11):2399–2434, 2006.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

D. Benielli, B. Bauvin, S. Koço, R. Huusari, C. Capponi, H. Kadri, and F. Laviolette. Toolbox for multimodal learn (scikit-multimodallearn). *Journal of Machine Learning Research*, 23(51):1–7, 2022.

M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining*, volume 4, pages 19–26, 2004.

L. Billard and E. Diday. From the statistics of data to the statistics of knowledge: Symbolic data analysis. *Journal of the American Statistical Association*, 98(462):470–487, 2003.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

H.-H. Bock and E. Diday. *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*. Springer Science & Business Media, 1999.

I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Science & Business Media, 2005.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. In *Proceedings of the 13th IEEE International Conference on Image Processing*, volume 3, pages III–161, 2005.

A. Boukerche, L. Zheng, and O. Alfandi. Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3):1–37, 2020.

J. R. Boyd. The essence of winning and losing. *Unpublished lecture notes*, 1996.

R. Brault, M. Heinonen, and F. Buc. Random Fourier features for operator-valued kernels. In *Asian Conference on Machine Learning*, pages 110–125. PMLR, 2016.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Routledge, 1984.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan. IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42: 335–359, 2008.

A. Caponnetto, C. A. Micchelli, M. Pontil, and Y. Ying. Universal multi-task kernels. *Journal of Machine Learning Research*, 9(52):1615–1646, 2008.

C. Carmeli, E. De Vito, and A. Toigo. Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem. *Analysis and Applications*, 4(04):377–408, 2006.

R. B. Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.

G. C. Cawley, G. J. Janacek, and N. L. Talbot. Generalised kernel machines. In *IEEE International Joint Conference on Neural Networks*, pages 1720–1725, 2007.

S. Chatterjee and A. S. Hadi. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, pages 379–393, 1986.

W. Chauvenet. *A Manual Of Spherical And Practical Astronomy Vol II.* J. B. Lippincott & Company, 1st edition, 1863.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

W. Chen and F. W. Samuelson. The average receiver operating characteristic curve in multireader multicase imaging studies. *The British Journal of Radiology*, 87(1040):20140016, 2014.

F. Cohen. Computer viruses: Theory and experiments. *Computers & Security*, 6(1):22–35, 1987.

R. D. Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18, 1977.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995.

K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems*, pages 553–560, 2003.

S. Das, B. L. Matthews, A. N. Srivastava, and N. C. Oza. Multiple kernel learning for heterogeneous anomaly detection: Algorithm and aviation safety case study. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 47–56, 2010.

J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240, 2006.

J. De Leeuw and W. J. Heiser. Convergence of correction matrix algorithms for multidimensional scaling. *Geometric Representations of Relational Data*, 36:735–752, 1977.

J. De Leeuw and P. Mair. Multidimensional scaling using majorization: SMACOF in R. *Journal of Statistical Software*, 31:1–30, 2009.

V. R. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems*, pages 112–119, 1994a.

V. R. de Sa. Minimizing disagreement for self-supervised classification. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 300–307, 1994b.

V. R. de Sa and D. H. Ballard. Category learning through multimodality sensing. *Neural Computation*, 10(5):1097–1117, 1998.

E. De Vito, V. Umanità, and S. Villa. An extension of Mercer theorem to matrix-valued measurable kernels. *Applied and Computational Harmonic Analysis*, 34(3):339–351, 2013.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

N. Deng, Y. Tian, and C. Zhang. *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. CRC Press, 2012.

W. J. Dixon. Processing data for outliers. *Biometrics*, 9(1):74–89, 1953.

H. Djidjev, G. Sandine, C. Storlie, and S. Vander Wiel. Graph based statistical analysis of network traffic. In *Proceedings of the 9th Workshop on Mining and Learning with Graphs*, 2011.

C. Drummond and R. C. Holte. Explicitly representing expected cost: An alternative to ROC representation. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 198—207, 2000.

D. P. Ellis and G. E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 1429–1432, 2007.

M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

J. Farquhar, D. Hardoon, H. Meng, J. S. Shawe-Taylor, and S. Szedmak. Two view learning: SVM-2K, theory and practice. In *Advances in Neural Information Processing Systems*, pages 355–362, 2006.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

C. Ferri, J. Hernández-Orallo, and P. A. Flach. A coherent interpretation of AUC as a measure of aggregated classification performance. In *International Conference on Machine Learning*, pages 657–664, 2011.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

B. N. Flury. Common principal components in K groups. *Journal of the American Statistical Association*, 79(388):892–898, 1984.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.

K. Fukumizu, F. R. Bach, and M. I. Jordan. Kernel dimension reduction in regression. *The Annals of Statistics*, 37(4):1871–1905, 2009.

E. Gandotra, D. Bansal, and S. Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 5:56–64, 2014.

E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proceedings of the 12th International Symposium on Graph Drawing*, pages 239–250, 2005.

T. Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

G. Giacinto, F. Roli, and L. Didaci. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recognition Letters*, 24(12):1795–1803, 2003.

D. Gibert, C. Mateu, and J. Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153:102526, 2020.

T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, R. A. Bridges, and Q. Chen. A survey of intrusion detection systems leveraging host data. *arXiv preprint arXiv:1805.06070*, 2018.

V. Gligorijević and N. Pržulj. Methods for biological data integration: Perspectives and challenges. *Journal of the Royal Society Interface*, 12(112), 2015.

J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871, 1971.

A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *International Conference on Algorithmic Learning Theory*, pages 63–77, 2005.

F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

D. J. Hand. Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123, 2009.

D. J. Hand and W. E. Henley. Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160(3):523–541, 1997.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

J. Hernández-Orallo, P. Flach, and C. Ferri. A unified view of performance metrics: Translating threshold choice into expected classification loss. *Journal of Machine Learning Research*, 13(91):2813–2869, 2012.

A. Hero, S. Kar, J. Moura, J. Neil, H. V. Poor, M. Turcotte, and B. Xi. Statistics and data science for cybersecurity. *Harvard Data Science Review*, 5(1), 2023.

K. Highnam, K. Arulkumaran, Z. Hanif, and N. R. Jennings. BETH dataset: Real cybersecurity data for anomaly detection research. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

J. Hogan and N. M. Adams. A study of data fusion for predicting novel activity in enterprise cyber-security. In *IEEE International Conference on Intelligence and Security Informatics*, pages 37–42, 2018.

J. Hogan and N. M. Adams. On averaging ROC curves. *Transactions on Machine Learning Research*, 2023.

J. Hogan and N. M. Adams. Multi-view generalised kernel machines for regression, classification and outlier detection. *Journal of Machine Learning Research*, under review 2023b.

D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression*, volume 1. John Wiley & Sons, 2013.

M. I. Hossain, V. Huroyan, S. Kobourov, and R. Navarrete. Multi-perspective, simultaneous embedding. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1569–1579, 2020.

Y. Huang, C. Du, Z. Xue, X. Chen, H. Zhao, and L. Huang. What makes multi-modal learning better than single (provably). *Advances in Neural Information Processing Systems*, 34:10944–10956, 2021.

R. Huusari, H. Kadri, and C. Capponi. General framework for multi-view metric learning. In *Linking and Mining Heterogeneous and Multi-view Data*, pages 265–294. Springer, 2019.

S. K. Izquierdo. *Multiview pattern recognition methods for data visualization, embedding and clustering*. PhD thesis, Universitat Politècnica de Catalunya (UPC), 2017.

P. Jaccard. The distribution of the flora in the Alpine zone. *New Phytologist*, 11(2):37–50, 1912.

S. Jajodia, P. Liu, V. Swarup, and C. Wang. *Cyber Situational Awareness*. Springer, 2009.

H. Janes and M. S. Pepe. Adjusting for covariate effects on classification accuracy using the covariate-adjusted receiver operating characteristic curve. *Biometrika*, 96(2):371–382, 2009.

P. Juszczak, N. M. Adams, D. J. Hand, C. Whitrow, and D. J. Weston. Off-the-peg and bespoke classifiers for fraud detection. *Computational Statistics & Data Analysis*, 52(9):4521–4532, 2008.

H. Kadri, S. Ayache, C. Capponi, S. Koço, F.-X. Dupé, and E. Morvant. The multi-task learning view of multimodal data. In *Asian Conference on Machine Learning*, pages 261–276, 2013.

H. Kadri, E. Duflos, P. Preux, S. Canu, A. Rakotomamonjy, and J. Audiffren. Operator-valued kernels for learning from functional response data. *Journal of Machine Learning Research*, 17(1):613–666, 2016.

T. Kamada, S. Kawai, et al. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

A. D. Kent. Cyber security data sources for dynamic network research. In *Dynamic Networks and Cyber-Security*, pages 37–65. World Scientific, 2016.

E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.

D. Kong and G. Yan. Discriminant malware distance learning on structural information for automated malware classification. In *Proceedings of the*

19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1357–1365, 2013.

M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Outlier detection in arbitrarily oriented subspaces. In *Proceedings of the 12th IEEE International Conference on Data Mining*, pages 379–388. IEEE, 2012.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964a.

J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964b.

W. J. Krzanowski and D. J. Hand. *ROC Curves for Continuous Data*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2009.

A. Kumar, P. Rai, and H. Daume. Co-regularized multi-view spectral clustering. *Advances in Neural Information Processing Systems*, 24:1413—1421, 2011.

C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.

G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004a.

G. R. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16): 2626–2635, 2004b.

D. J. Lawson, P. Rubin-Delanchy, N. Heard, and N. M. Adams. Statistical frameworks for detecting tunnelling in cyber defence using big data. In

IEEE Joint Intelligence and Security Informatics Conference, pages 248–251, 2014.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *Advances in Neural Information Processing Systems*, 19:801—808, 2006.

E. Lesaffre and A. Albert. Multiple-group logistic regression diagnostics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 38 (3):425–440, 1989.

Y. Li, M. Yang, and Z. Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):1863––1883, 2019.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the 13th European Conference on Computer Vision*, pages 740–755, 2014.

J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

B. Long, P. S. Yu, and Z. Zhang. A general model for multiple view unsupervised learning. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 822–833, 2008.

Q. Lu and L. Getoor. Link-based classification. In *International Conference on Machine Learning*, pages 496–503, 2003.

Z. Ma, A. Cardinal-Stakenas, Y. Park, M. W. Trosset, and C. E. Priebe. Dimensionality reduction on the Cartesian product of embeddings of multiple dissimilarity matrices. *Journal of Classification*, 27(3):307—-321, 2010.

S. A. Macskassy and F. Provost. Confidence bands for ROC curves: Methods and an empirical study. In *Proceedings of the 1st Workshop on ROC Analysis in AI*, pages 61–70, 2004.

P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Science of India*, volume 12, pages 49–55, 1936.

G. L. Marcialis and F. Roli. Fusion of LDA and PCA for face verification. In *International Workshop on Biometric Authentication*, pages 30–37, 2002.

A. Marcos Alvarez, M. Yamada, A. Kimura, and T. Iwata. Clustering-based anomaly detection in multi-view data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 1545–1548, 2013.

B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*, volume 8, pages 18–25, 2015.

J. Mercer. 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, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, 116:374–388, 1976.

C. E. Metz. Some practical issues of experimental design and data analysis in radiological ROC studies. *Investigative Radiology*, 24(3):234–245, 1989.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005.

G. Michailidis. Challenges for anomaly detection in large-scale cyber-physical systems. *Harvard Data Science Review*, 5(1), 2023.

H. Q. Minh and V. Sindhwani. Vector-valued manifold regularization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 57–64, 2011.

H. Q. Minh, L. Bazzani, and V. Murino. A unifying framework for vector-valued manifold regularization and multi-view learning. In *International Conference on Machine Learning*, pages 100–108, 2013.

H. Q. Minh, L. Bazzani, and V. Murino. A unifying framework in vector-valued reproducing kernel Hilbert spaces for manifold regularization and co-regularized multi-view learning. *Journal of Machine Learning Research*, 17(1):769–840, 2016.

M. Morgan, J. Sexton, J. Neil, A. Ricciardi, and J. Theimer. Network attacks and the data they affect. In *Dynamic Networks and Cyber-Security*, pages 1–36. World Scientific, 2016.

Y. Mroueh, T. Poggio, L. Rosasco, and J.-J. Slotine. Multiclass learning with simplex coding. In *Advances in Neural Information Processing Systems*, pages 2789–2797, 2012.

B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, 1994.

G. Namata, B. London, L. Getoor, B. Huang, and U. Edu. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, volume 8, 2012.

J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, 55 (4):403–414, 2013.

J. A. Nelder and R. W. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (Coil-100). Technical report, CUCS-006-96, 1996.

S. M. T. Nezhad, M. Nazari, and E. A. Gharavol. A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks. *IEEE Communications Letters*, 20(4):700–703, 2016.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning*, pages 689–696, 2011.

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, volume 5, pages 86–93, 2000.

T. Niu, S. Zhu, L. Pang, and A. E. Saddik. Sentiment analysis on multi-view social data. In *International Conference on Multimedia Modeling*, pages 15–27, 2016.

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583): 607–609, 1996.

C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *International Conference on Machine Learning*, pages 81–88, 2004.

W. Ouyang, X. Chu, and X. Wang. Multi-source deep learning for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2329–2336, 2014.

S. Panjwani, S. Tan, K. M. Jarrin, and M. Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, pages 602–611, 2005.

B. J. Parker, S. Günter, and J. Bedo. Stratification bias in low signal microarray studies. *BMC Bioinformatics*, 8(1):1–16, 2007.

E. Parzen. Extraction and detection problems and reproducing kernel Hilbert spaces. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(1):35–62, 1962.

K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, number 3, pages 61–74, 1999.

G. Pouliot. Equivalence of multicategory SVM and simplex cone SVM: Fast computations and statistical theory. In *International Conference on Machine Learning*, pages 4130–4137, 2018.

D. Pregibon. Logistic regression diagnostics. *The Annals of Statistics*, 9(4): 705–724, 1981.

F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 43—48, 1997.

F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *International Conference on Machine Learning*, pages 445–453, 1998.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning

transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20:1177–1184, 2007.

E. Riddle-Workman, M. Evangelou, and N. M. Adams. Adaptive anomaly detection on network data streams. In *IEEE International Conference on Intelligence and Security Informatics*, pages 19–24, 2018.

K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125, 2008.

R. Rifkin, G. Yeo, and T. Poggio. Regularized least-squares classification. In *Advances in Learning Theory: Methods, Model and Applications. NATO Science Series III: Computer and Systems Sciences*, volume 190, pages 131–154. IOS Press, 2003.

A. Ross and A. Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24(13):2115–2125, 2003.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

F. Sanna Passino, N. M. Adams, E. A. Cohen, M. Evangelou, and N. A. Heard. Statistical cybersecurity: A brief discussion of challenges, data structures, and future directions. *Harvard Data Science Review*, 5(1), 2023a.

F. Sanna Passino, A. Mantziou, D. Ghani, P. Thiede, R. Bevington, and N. A. Heard. Unsupervised attack pattern detection in honeypot data using Bayesian topic modelling. *arXiv preprint arXiv:2301.02505*, 2023b.

E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1331–1334, 1997.

B. Schölkopf. *Support vector learning*. PhD thesis, Oldenbourg München, Germany, 1997.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 38–49, 2000.

H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to Information Retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

A. Serra, M. Fratello, V. Fortino, G. Raiconi, R. Tagliaferri, and D. Greco. MVDA: A multi-view genomic data integration methodology. *BMC Bioinformatics*, 16(1):1–13, 2015.

M. B. Shah and B. C. Rana. *Engineering Drawing*. Pearson Education India, 2009.

R. N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, 1962.

V. Sindhwani and D. S. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *International Conference on Machine Learning*, pages 976–983, 2008.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*, pages 74–79, 2005.

T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. ROCR: Visualizing classifier performance in R. *Bioinformatics*, 21(20):3940–3941, 2005.

T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. Package 'ROCR'. *Visualizing the Performance of Scoring Classifiers*, pages 1–14, 2015.

A. Skaron, K. Li, and X.-H. Zhou. Statistical methods for MRMC ROC studies. *Academic Radiology*, 19(12):1499–1507, 2012.

R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy*, pages 305–316, 2010.

K. Srinivasan, K. Raman, J. Chen, M. Bendersky, and M. Najork. WIT: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2443–2449, 2021.

N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep Boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2222–2230, 2012.

W. Stafford Noble. Support vector machine applications in computational biology. In *Kernel Methods in Computational Biology*, pages 71–92. MIT Press, 2004.

F.-R. Stöter, S. Chakrabarty, B. Edler, and E. A. Habets. Classification vs. regression in supervised learning for single channel speaker count estimation. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 436–440, 2018.

I. Subramanian, S. Verma, S. Kumar, A. Jere, and K. Anamika. Multi-omics data integration, interpretation, and its application. *Bioinformatics and Biology Insights*, 14:1–24, 2020.

J. A. Swets and R. M. Pickett. *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory*. Academic Press, New York, 1982.

J. A. Swets, R. M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, 283(4):82–87, 2000.

R. Talpade, G. Kim, and S. Khurana. NOMAD: Traffic-based network monitoring framework for anomaly detection. In *Proceedings of the IEEE International Symposium on Computers and Communications*, pages 442–451, 1999.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): 2319–2323, 2000.

W. S. Torgerson. *Theory and Methods of Scaling*. Wiley, 1958.

N. T. Trendafilov. Stepwise estimation of common principal components. *Computational Statistics & Data Analysis*, 54(12):3446–3457, 2010.

M. J. M. Turcotte, A. D. Kent, and C. Hash. Unified host and network data set. In *Data Science for Cyber-Security*, pages 1–22. World Scientific, 2018.

M. van Breukelen, R. P. Duin, D. M. Tax, and J. Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2013.

V. N. Vapnik and A. Y. Chervonenkis. On the method of ordered risk minimization, (I and II). *Automation and Remote Control*, 34:1226–1235 and 1403–1412, 1974.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, 2008.

J. L. Wayman. Technical testing and evaluation of biometric identification devices. *Biometrics: Personal Identification in Networked Society*, 479: 345–368, 1999.

A. R. Webb. *Statistical Pattern Recognition*. John Wiley & Sons, 2003.

M. Whitehouse, M. Evangelou, and N. M. Adams. Activity-based temporal anomaly detection in enterprise-cyber security. In *IEEE International Conference on Intelligence and Security Informatics*, pages 248–250, 2016.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 13:682–688, 2000.

I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 3nd edition, 2011.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

Y. Yang, J. Song, Z. Huang, Z. Ma, N. Sebe, and A. G. Hauptmann. Multi-feature fusion via hierarchical regression for multimedia analysis. *IEEE Transactions on Multimedia*, 15(3):572–581, 2012.

N. Ye, S. M. Emran, Q. Chen, and S. Vilbert. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on Computers*, 51(7):810–820, 2002.

Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar. A survey on malware detection using data mining techniques. *ACM Computing Surveys*, 50(3):1–40, 2017.

H. Zhang, Y. Xu, and Q. Zhang. Refinement of operator-valued reproducing kernels. *Journal of Machine Learning Research*, 13(1):91–136, 2012.

L. Zhang, Y. Xie, L. Xidao, and X. Zhang. Multi-source heterogeneous data fusion. In *Proceedings of the IEEE International Conference on Artificial Intelligence and Big Data*, pages 47–51, 2018.

H. Zhao, H. Liu, Z. Ding, and Y. Fu. Consensus regularized multi-view outlier detection. *IEEE Transactions on Image Processing*, 27(1):236–248, 2017a.

J. Zhao, X. Xie, X. Xu, and S. Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017b.

G. Zhong, L.-N. Wang, X. Ling, and J. Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265–278, 2016.

X.-H. Zhou, D. K. McClish, and N. A. Obuchowski. *Statistical Methods in Diagnostic Medicine*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2009.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*, volume 14, pages 1081–1088, 2001.

M. Žitnik and B. Zupan. Data fusion by matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):41–53, 2015.