

Imperial College London

Department of Electrical and Electronic Engineering

Efficient Inference and Sampling Methods for Probabilistic Models

Kiarash Shaloudegi

May 2019

Supervisor: Dr. András György

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London
and the Diploma of Imperial College London

Abstract

Probabilistic modeling is the main practical approach for designing systems that learn from data, and has numerous applications in different fields of science and engineering. Given a probabilistic model of a system and its associated parameters, general queries of interest about the system can be answered by performing inference on the distribution. Considering the importance of inference methods for machine learning, our goal in this dissertation is to develop efficient approximate inference methods for some practical problems. For this purpose, the following two problems are considered:

First we consider the problem of non-intrusive home-appliance load monitoring which can be modeled as factorial hidden Markov model (FHMM). FHMMs have applications in several fields such as voice recognition, home appliance monitoring, computer vision, and fault detection. By employing FHMMs, the load-disaggregation problem becomes an FHMM inference problem described by a non-convex quadratic integer optimization problem. In this thesis we present a computationally efficient method that finds an approximate solution to this problem by combining convex semidefinite relaxations, randomized rounding, as well as a scalable ADMM method. Simulation results from both synthetic and real-world datasets demonstrate the superiority of our method compared to the previous state-of-the-art solutions.

Markov chain Monte Carlo (MCMC) methods are widely used in machine learning. One of the major problems with MCMC is the question of how to design chains that mix fast over the whole state space; in particular, how to select the parameters of an MCMC algorithm. In this thesis we take a different approach and, similarly to parallel MCMC methods, instead of trying to find a single chain that samples from the whole distribution, we combine samples from several chains run in parallel, each exploring only parts of the state space. The chains are prioritized based on the kernel Stein discrepancy. The samples from the independent chains are combined using a novel technique for estimating the probability of different regions of the sample space. Experimental results demonstrate that the proposed algorithm provides significant

speed-ups in different sampling problems.

Declaration of Originality

I hereby certify that this thesis is the result of my own work under the guidance of my Ph.D. advisor, Dr. András György. Any ideas or quotations from the work of other people are appropriately referenced.

Imperial College London
London, United Kingdom
May 2019

Kiarash Shaloudegi

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Acknowledgements

First of all, I would like to thank my supervisor, András György, for his kindness, encouragement, honest feedback, and invaluable advice. Through these years, he set a real example of academic and research integrity for me. Also, I would like to thank my former co-supervisor at the University of Alberta, Csaba Szepesvári, for his guidance, patience, and support. Through his manner, he taught me a great deal about the qualities of a true teacher and a great leader. Without their help and support, my Ph.D. studies would have been much more difficult.

I would like to thank Osmar R. Zaiane who initially introduced me to the field of machine learning and computer science, and I had the honor of being a member of his research team for roughly a year. Also, I would like to thank my close friend and colleague, Pooria Joulani. I benefited from numerous insightful discussions with him and his technical advice. I would like to thank Firas Hamze for his constructive comments and discussions about different sampling methods and Tor Lattimore for his comments on the MCMC paper.

During my studies, I was lucky enough to know and become friends with many great people who made my Ph.D. a treasured experience. So, I would like to acknowledge them for their help and support: Mostafa Tabatabaei, Jean-Paul Prévost, Muhammad Zaid Hameed, Samuel O Somuyiwa, David Kurka, Elif Ceran, Junlin Zhao, Andrea Paudice, Ming Dong, Giulio Giaconi, Sreejith Sreekumar, Qianqian Yang, Mohammad Ajallooeian, Pengfei Gao, Borzoo Rassouli, Mohammad Mohammadi Amiri, and many other friends and colleagues at the University of Alberta and Imperial College London.

My everlasting gratitude goes to my parents. Their unconditional love, encouragement, and support have always been my true source of inspiration in life. My endless thanks to my brother and best friend, Koushyar, who helped me in too many ways to count. At the end, my deepest loves go to my beloved wife, Saha. This dissertation would not exist if it was not for her love, support, trust, and her many sacrifices.

*To my parents, Nader and Sophia,
to my wife, Saha,
and to my brother, Koushyar.*

Contents

List of Figures	11
List of Tables	15
List of Acronyms	16
1 Introduction	17
1.1 Objectives and Contributions	18
1.1.1 Non-Intrusive Appliance Load Monitoring	19
1.1.2 Adaptive MCMC	20
2 Scalable FHMM Inference Algorithm for Energy Disaggregation	23
2.1 Introduction	23
2.1.1 Notation	26
2.2 System Model	27
2.3 SDP Relaxation and Randomized Rounding	29
2.3.1 Approximate Solutions for Integer Quadratic Programming	29
2.4 An Efficient Algorithm for Inference in FHMMs	31
2.4.1 Scalable Solutions for Large SDP Problems	32
2.4.1.1 Background: Alternating Direction Method of Multipliers (ADMM)	33
2.4.1.2 ADMM for SDP Problems	34
2.4.2 Randomized Rounding	37
2.5 Learning the Model	38
2.5.1 A Spectral Learning Method for Discrete-Output HMMs	39
2.5.2 The QP Learning Method	42
2.5.2.1 Learning Discrete-Output HMM	42
2.5.2.2 Learning Continuous-Output HMM	43
2.5.3 Learning HMMs for NILM Application	44
2.6 Unmodeled Appliances	44
2.7 Experimental Results	45
2.7.1 Learning the Parameters of an HMM	45
2.7.2 Inference on Synthetic Data	46
2.7.3 Real Data	48
2.8 Conclusions and Future Work	50
Appendix 2.A Discussion of the Derivation of Kolter & Jaakkola (2012) in the Presence of the “Generic Model”	53
3 Adaptive MCMC via Combining Local Samplers	54
3.1 Introduction	54
3.1.1 Approach and Challenges	59

3.2	Background: Markov Chain Monte Carlo Methods	61
3.2.1	The Metropolis-Hastings Algorithm	66
3.2.2	Langevin Algorithms	67
3.2.3	Hamiltonian Monte Carlo	69
3.2.4	Sequential Monte Carlo Sampler Algorithms	72
3.2.5	Parallel Tempering	74
3.3	Measuring Sample Quality	75
3.3.1	Stein Discrepancy	76
3.3.2	Kernel Stein Discrepancy	77
3.4	Background: Multi-Armed Bandit Problems	81
3.5	Sequential Selection of Samplers	84
3.5.1	Mixing Samplers	84
3.5.2	Locally Mixing Samplers	86
3.5.3	Weight Estimation	89
3.6	The Final Algorithm	91
3.7	Experiments	92
3.7.1	Block-Diagonal Approximation of the KSD Measure	94
3.7.2	Unimodal Target Distributions with Multiple Samplers	95
3.7.3	Separated Modes with One Sampler for Each Mode	98
3.7.4	Weight Estimation	100
3.7.5	The General Case: Unknown Number of Modes with Multiple Chains	102
3.7.6	Comparison with Parallel MCMC Methods	105
3.7.7	Comparing the Components of KSD-MCMC-WR: Bandit Algo- rithms and Clustering-Reweighting	108
3.7.8	Sensor Network Localization	112
3.8	Conclusions and Future Work	113
4	Summary and Future Work	115

List of Figures

2.1	Graphical representation of FHMMs.	27
2.2	Sample power consumption of a stove from the REDD dataset (Kolter & Johnson, 2011). Each state i of the appliance is associated with an approximately constant power level μ_i ; in this case $\mu = [1050, 220, 1270, 0]^\top$. The power consumption at time t is modeled as $y_t = x_t^\top \mu + n_t$ where x_t is the state indicator vector (e.g., $x_{350} = [0, 1, 0, 0]^\top$) and n_t is the noise at time t	28
2.3	The mean squared-error in estimating the entires of the emission matrix O (top) and the transition matrix P (bottom) as a function of the number of observations.	46
2.4	Disaggregation error varying the number of HMMs. The number of states is fixed and equal to 3. The length of the observation signal is 1000 and the results are averaged over 100 random realizations.	48
2.5	Disaggregation error varying the number of states. The number of appliances is fixed and equal to 5. The length of the observation signal is 1000 and the results are averaged over 100 random realizations.	48
3.1	Simulating a MH algorithm with different proposal functions. The ellipsoids represent the contours of the target distribution. The same number of samples are taken from both proposals. The left figure corresponds to a MH algorithm with small moves, while the right figure corresponds to a MH algorithm with large moves.	57
3.2	Block-diagonal approximation of the KSD measure for kernel width $h = 1$	96
3.3	The average running time for different KSD based method.	97
3.4	Unimodal case, Gaussian target distribution: MSE for different sample sizes with MH samplers in 2-dimensions ($d = 2$) with batch size 10 (left) and batch size 100 (right). The dashed lines without labels show the performance of the different MH samplers. The total number of samples is 5000 in each case.	98
3.5	Unimodal case, Gaussian target distribution: MSE versus dimension d with MH samplers and batch size 10(left) and batch size 100 (right). The dashed lines without labels show the performance of the different MH samplers. The total number of samples is 5000 in each case.	98
3.6	Unimodal case, Gaussian target distribution: MSE for different sample sizes with MALA samplers in 2-dimensions ($d = 2$) with batch size 10 (left) and batch size 100 (right). The dashed lines without labels show the performance of the different MALA samplers. The total number of samples is 5000 in each case.	99
3.7	Unimodal case, Gaussian target distribution: MSE versus dimension d with MALA samplers and batch size 10 (left) and batch size 100 (right). The dashed lines without labels show the performance of the different MALA samplers. The total number of samples is 5000 in each case.	99

- 3.8 Multimodal distributions (Gauss mixtures): MSE for different sample sizes when the samplers are MH when the weight of each mode is know in advance (left); and when the weights are estimated with our proposed weight estimation method based on Rényi entropy with $\alpha = 0.95$ (right). The samples are taken from the following unnormalized distribution $p(x) = \sum_{i=1}^3 \beta_i \exp \{(x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i)\}$ where $\beta = [0.5, 0.3, 0.2]$, $\Sigma_i = \sigma_i I$ with $\sigma = [0.9, 0.4, 0.5]$ and $\mu_1 = [6, 6]^\top$, $\mu_2 = [-6, 6]$ and $\mu_3 = [0, -6]$. On the right figure one can observe that the error slightly goes up after a sharp decline for all the methods (after about 5000 samples). This is due to a single chain that visited some areas of low probability that affected the weight estimation. The error starts to decline again when the chain returns to a high probability area. 101
- 3.9 Multimodal distributions (Gauss mixtures): MSE for different sample sizes for the same target distribution as in Figure 3.8 with MALA base samplers when the weight of each mode is know in advance (left); and when the weights are estimated with our proposed weight estimation method based on Rényi entropy with $\alpha = 0.95$ (right). 101
- 3.10 Multimodal distributions (Gauss mixtures): Mean squared error of different weight estimation methods for the setup in Figure 3.8 when the samplers receive equal budget. “Gaussian” refers to considering a Gaussian distribution for each mode and calculating its weight by estimating its covariance matrix, while “Rényi” refers to our proposed weight-estimation method based on Rényi entropy. Results for MH base samplers are shown on the left, while for MALA samplers on the right. . . . 102
- 3.11 Unknown number of modes: Mean squared error of different weight estimation methods for a multimodal distribution in 2-dimensions ($d=2$) when combining 10 randomly initialized samplers. The samples are taken from a 2-dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode is selected uniformly at random from $[-5, 5]^2$ and the variance of each component is selected randomly from $[0.2, 1]$. “Gaussian” refers to considering a Gaussian distribution for each mode and calculating its weight by estimating its covariance matrix, while “Rényi” refers to our proposed weight-estimation method based on Rényi entropy with $\alpha = 0.99$. Results for MH base samplers are shown on the left, while for MALA samplers on the right. 103
- 3.12 Unknown number of modes: Mean squared error of different weight estimation methods for a multimodal distribution in d -dimensions when combining 10 randomly initialized NUTS samplers. The samples are taken from a d -dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode is selected uniformly at random from $[-5, 5]^d$ and the variance of each component is selected randomly from $[0.5, 1]$. “Gaussian” refers to considering a Gaussian distribution for each mode and calculating its weight by estimating its covariance matrix, while “Rényi” refers to our proposed weight-estimation method based on Rényi entropy with $\alpha = 0.99$ 103

- 3.13 Unknown number of modes: MSE for different sample sizes when combining 10 randomly initialized MH samplers. The samples are taken from a 2-dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode is selected uniformly at random from $[-5, 5]^2$ and the variance of each component is selected randomly from $[0.2, 1]$. Two representative cases are shown: in the top row the modes are far from each other, in the bottom row the modes are close. The left column corresponds to batch size 10, while it is 100 for the right column. Rényi entropy with $\alpha = 0.99$ is used for weight estimation. 105
- 3.14 MSE for different sample sizes with separated modes: KSD-MCMC-WR with 10 randomly initialized NUTS samplers for an unknown number of modes. The batch size on the left figure is 10 and on the right figure is 100. The dashed green lines show the results for the individual NUTS samplers. Rényi entropy with $\alpha = 0.99$ is used for weight estimation. The problem setup is the same as the one in Figure 3.13. 106
- 3.15 Running KSD-MCMC-WR for a multivariate standard normal distribution with 10 NUTS instances started with different (random) initializations. The batch size is 10 on the left and 100 on the right. 106
- 3.16 Unknown number of modes: MSE for different sample sizes when combining 10 randomly initialized MALA samplers for a 2-dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode was selected uniformly at random from $[-5, 5]^2$ and the variance of each component was selected uniformly from $[0.2, 1]$. SMC and PT were run separately for every parameter setting of MALA (10 different parameters) with the best, worst and average values reported, while KSD-MCMC-WR combined the same samplers. In addition, the results for KSD-MCMC-WR with 10 randomly initialized NUTS base samplers were included for completeness. Batch size for KSD-MCMC-WR is 10 on the left figure, 100 on the right. Each curve is obtained by averaging over 100 runs with different random initializations of the MALA/NUTS samplers (with the same parameters). 108
- 3.17 Unknown number of modes: MSE for different dimensions when using KSD-MCMC-WR with 10 randomly initialized NUTS samplers for a d -dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode was selected uniformly at random from $[-5, 5]^d$ and the variance of each component was selected randomly from $[0.5, 1]$. SMC and PT were run separately for every parameter setting of MALA (10 different parameters) with the best, worst and average values reported. Batch size for KSD-MCMC-WR is 10 on the left figure, 100 on the right. Each curve is obtained by averaging over 100 runs with different random initializations of the MALA/NUTS samplers (for the same parameters) 109

- 3.18 Changing the KSD kernel width h : MSE for different dimensions when using KSD-MCMC-WR with 10 randomly initialized NUTS samplers (the same setting as in Figure 3.17). Batch size for KSD-MCMC-WR is 10 on the left figures, 100 on the right; the performance of the UCB1 variant is shown in the top row, while that of the ε -greedy version in the bottom. Each curve is obtained by averaging over 100 runs with different initializations. 110
- 3.19 MSE for different number of NUTS samplers as a function of the number of samplers for a 2-dimensional Gaussian mixture model with 20 isotropic modes. The mean of each mode was selected uniformly at random from $[-10, 10]^2$ and the variance of each component was selected uniformly from $[0.2, 1]$. Two representative cases are shown: in the top row the modes are far from each other, in the bottom row some modes are close to each other. Batch size for KSD-MCMC-WR is 10 on the left figure and 100 on the right. Each curve is obtained by averaging over 100 runs with different random initializations of the NUTS samplers; the total number of samples is 20000 in each case. 111
- 3.20 Running KSD-MCMC-WR for the sensor network localization problem with 10 NUTS instances started with different (random) initialization. The dashed green lines show the results for the individual NUTS samplers. Rényi entropy with $\alpha = 0.99$ is used for weight estimation. The batch size is 10 on the left and 100 on the right. 113

List of Tables

2.1	Disaggregation performance on the REDD dataset: precision/recall and the error in the assigned power to each appliance as a percentage of the total power (for ADMM-RR we report averages over 5 runs and the corresponding maximum deviations). For calculating the weighted averages, the share of each appliance in the total energy consumption is considered as its weight.	51
2.2	Disaggregation performance on the REDD dataset: the error in the assigned power to each appliance as a percentage of the total power (for ADMM-RR we report averages over 5 runs and the corresponding maximum deviations). For calculating the weighted averages, the share of each appliance in the total energy consumption is considered as its weight.	51
2.3	Precision/recall of disaggregation on the UK-DALE dataset.	52
3.1	The fraction of times the samplers are ranked in the same order for different block sizes as for block size 2000 (regarded as the “ground truth”) for different kernel widths h	95

List of Acronyms

ADMM	Alternating Direction Method of Multipliers
a.s.	Almost Surely
EM	Expectation Maximization
ESS	Effective Sample Size
FHMM	Factorial Hidden Markov Model
HMC	Hamiltonian Monte Carlo
HMM	Hidden Markov Model
i.i.d.	Independent and Identically Distributed
IPM	Interior-Point Method
KL	Kullback–Leibler
KSD	Kernel Stein Discrepancy
MALA	Metropolis-Adjusted Langevin Algorithm
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MH	Metropolis-Hastings
NILM	Non-Intrusive Load Monitoring
NUTS	No-U-Turn Sampler
PT	Parallel Tempering
QP	Quadratic Program
RKHS	Reproducing Kernel Hilbert Space
RR	Randomized Rounding
SDP	Semi-Definite Programming
SMC	Sequential Monte Carlo
TV	Total Variation
UCB	Upper Confidence Bound

Chapter 1

Introduction

Probabilistic modeling is the main practical approach for designing systems that learn from data (Ghahramani, 2015). The probabilistic framework plays a fundamental role in different fields such as robotics, communication, control, machine learning and artificial intelligence (Barber, 2012; Ghahramani, 2015; Jordan, 2004). In all of these examples, there is a complex system of interrelated variables and subsystems that we want to develop a plausible model about their behavior to control them, or to predict their future behavior. To achieve this goal, we need to model uncertainties associated with these systems. These uncertainty come from either our imperfect knowledge of the environment or noise in the observations, and probabilistic modeling provides a framework to model them by considering the uncertainties as random variables and forming a joint distribution over the space of all possible values of the variables (Barber, 2012; Koller & Friedman, 2009).

Graphical models have emerged as a general framework for implementing probabilistic models (Bishop, 2006). They provide a unified platform for studying a wide class of probabilistic models and their associated inference algorithms. A graphical model consists of a graph that can compactly represent a joint distribution. In such a graph, a node represents a random variable or group of random variables, and an edge between two nodes shows conditional dependence among their variables. Causal relationship (i.e., conditional relationship) among variables can be also modeled through directed graphs. Graphical models help us encode our prior knowledge on how variables affect each other in a clear way with a graph structure. In this way, seemingly different inference (i.e., learning) algorithms that are developed in different communities can be unified as special cases of a generic algorithm (Barber, 2012).

In general, there are two fundamental steps in problem environments using graphical/probabilistic models: modeling and inference (Barber, 2012). Modeling concerns with identifying some relevant variables of the complex distribution, exploiting the structure among them, and finally describing their interactions through a graph structure. Inference is the task of answering the desired queries about a system given its model and some observed data. Indeed, most of the tasks in machine learning (estimation, prediction, tests, variable selection, etc.) can be formulated as making inference about latent variables or missing data given observations of the system (Ghahramani, 2015; Robert & Casella, 2005). Unfortunately, performing exact inference for most real-world applications is computationally intractable, which has resulted in considerable efforts to develop approximate solutions. For example, calculating the posterior distribution of latent variables given observations of a system and/or maximizing it are NP-hard problems in a general Bayesian network (Shimony, 1994). In all of these computationally intractable cases, practical algorithms are developed for specific problems by sacrificing one or more algorithmic properties such as *generality*, *optimality*, and *guarantee of worst-case complexity* (Bakir et al., 2007; Nowozin & Lampert, 2011).

1.1 Objectives and Contributions

Given the importance of inference methods for machine learning in general, and graphical models in particular, our goal is to develop efficient inference methods for some practical problems. To do so, we consider two general problems in this thesis: The first problem is concerned with developing an efficient inference method for factorial hidden Markov models (FHMMs), while the second problem is focused on developing adaptive Markov chain Monte Carlo (MCMC) sampling methods. FHMMs have applications in several fields such as voice recognition (Rennie et al., 2009a,b), home appliance monitoring (Kolter & Jaakkola, 2012), computer vision (Oliver et al., 2000), and fault detection (Kodali et al., 2013). MCMC methods also have numerous applications in various fields of science and engineering such as cosmology (Lewis & Bridle, 2002), medicine (Baele et al., 2017; Cauchemez et al., 2004), finance (Johannes

& Polson, 2010), or multiple target tracking (Kokkala & Särkkä, 2015).¹

1.1.1 Non-Intrusive Appliance Load Monitoring

Non-intrusive appliance load monitoring (NILM) is a technique that uses data from existing and previously installed utility smart meters to separate individual energy consumptions of different appliances from the total measured power in households. Modern NILM methods use advanced pattern recognition and machine learning algorithms to achieve their goal. NILM provides feedback about consumption patterns of each load; this feedback educates consumers, helps them to identify potential areas for energy savings, and increases their willingness to invest in more efficient products. In addition, NILM provides the opportunity to the users to participate in load shifting programs offered by utility providers to take full advantage of *time-of-use* pricing, and allows remote monitoring of potentially hazardous appliances (e.g., alert a user if the oven was accidentally left on); see (Kelly & Knottenbelt, 2015b; Yu et al., 2016) for more details about the benefits of NILM.

In the literature, the most successful NILM algorithms are based on modeling the electrical system of households by hidden Markov models (HMMs) (Kim et al., 2011; Kolter & Jaakkola, 2012). The underlying assumption in these models is that the power consumption of each individual appliance is generated by a discrete state Markov process such that the power used by an appliance in each state remains around a constant level. The individual power consumption levels are then added together, and this is the combined power signal that is observed at the meter. The resulting model is called a factored HMM (FHMM), whose factors are the individual HMMs. The problem with using FHMMs for NILM is that doing exact inference—finding the most probable sequence of hidden states given the observations—is computationally intractable, since typical FHMM inference problems for NILM involve millions of decision variables (i.e., hidden states).

The state-of-the-art NILM algorithm prior to our method was the algorithm of Kolter & Jaakkola (2012) which defines the NILM problem as inference in FHMMs. To address

¹ MCMC has been named as one of the top 10 most influential algorithms of the 20th century in the fields of science and engineering (Dongarra & Sullivan, 2000).

the difficulty of FHMM inference, Kolter & Jaakkola (2012) formulated the inference problem as a quadratic integer optimization problem and solved it approximately by relaxing the integrality constraints and using quadratic programming. In this thesis we take a more principled approach, better suited to integer programming problems. We find an approximate optimum by combining convex semidefinite relaxations and randomized rounding. The resulting optimization problem is a large scale semidefinite program which cannot be solved with existing SDP solvers. Therefore, a scalable special-purpose SDP solver is developed based on the ADMM algorithm (Boyd et al., 2011) which exploits the special structure of the problem at hand. Simulation results both on synthetic and real-world datasets demonstrate the superiority of our method. We expect our method to be useful in solving other FHMM inference problems, as well as in large scale integer quadratic programming. We also developed a novel variant of spectral learning algorithms by combining two existing spectral learning algorithms for learning HMM parameters. The experimental results show the superiority of the proposed combination method for learning HMM parameters. The results on this topic have been published in

- Shaloudegi, K.; György, A.; Szepesvári, C.; Xu, W., “SDP Relaxation with Randomized Rounding for Energy Disaggregation,” In Advances in Neural Information Processing Systems (NIPS), Spain, 2016 (oral presentation, acceptance rate 2%).

1.1.2 Adaptive MCMC

In the second problem, our goal is to estimate expectations of the form $\mathbb{E}_P[f(X)] = \int_{\mathcal{X}} f(x)p(x)dx$ for some complicated target distribution P with density p over a set $\mathcal{X} \subset \mathbb{R}^d$ for some function $f : \mathcal{X} \rightarrow \mathbb{R}$. Such integrals arise, among others, in Bayesian inference and maximum likelihood estimation problems from statistics, in econometrics, physics and computing science (Andrieu et al., 2003; Brooks et al., 2011; Gorham & Mackey, 2015). Unfortunately, calculating these integrals is usually intractable: One of the most popular methods to compute them in practice is to estimate them by using Markov Chain Monte Carlo (MCMC) methods. MCMC approximates the integral by

$\mathbb{E}_{Q_n}[f(Z)] = \sum_{i=1}^n f(x_i)q_n(x_i)$, where $\{x_i\}_{i=1}^n$ is a set of samples generated through a Markov chain and weighted according to the weights q_n . MCMCs have asymptotic performance guarantees, which means Q_n , the distribution of the weighted sample, converges to P .

The challenging problem in designing MCMC methods is to ensure that the distribution of the samples Q_n converges to P fast, and, in practice, some domain specific knowledge has to be used in the design of their Markov chains to achieve fast convergence (Andrieu et al., 2003), see Section 3.1 for more details. Finding a fast-converging method is crucial since using samples whose empirical distribution is far from the target (i.e., not converged yet) can lead to random results. Also, especially in big data applications, taking samples may be computationally expensive, so we want the smallest number of samples that best represent the target distribution. Lots of work have been done to speed up MCMC-type methods by designing Markov chains that converge fast (Atchadé et al., 2011; Bédard, 2008; Hoffman & Gelman, 2014; Mbalawata et al., 2015; Sejdinovic et al., 2014; Łatuszyński et al., 2013); for more details, see Section 3.1.

In this problem we take a different approach from previous works and, similarly to parallel MCMC methods, instead of trying to find a single chain that samples from the whole distribution, we combine samples from several chains run in parallel, each exploring only parts of the state space (e.g., a few modes only). The chains are prioritized based on the kernel Stein discrepancy, which provides a good measure of performance locally. Kernel Stein discrepancy is an empirical quality measure based on Stein’s method that can monitor “sample bias” in mean-square sense (Gorham & Mackey, 2015). The samples from the independent chains are combined using a novel technique for estimating the probability of different regions of the sample space. The proposed weight estimation technique is based on estimating Rényi entropy. For estimating the entropy, we use algorithms which are called *direct* entropy estimators; which are not built on density estimators and skirt the difficult intermediate step of multivariate density estimation required by other methods (Pál et al., 2010).

Experimental results demonstrate that the proposed adaptive sampler may provide significant speedups in different sampling problems. Most importantly, when combined

with the state-of-the-art NUTS algorithm as the base MCMC sampler, our method remained competitive with NUTS on sampling from unimodal distributions, while significantly outperforming state-of-the-art competitors on synthetic multimodal problems as well as on a challenging sensor localization task. The results on this topic have been published as

- Shaloudegi, K.; György, A., “Adaptive MCMC via Combining Local Samplers,” In International Conference on Artificial Intelligence and Statistics (AISTATS), Japan, 2019.

Chapter 2

Scalable FHMM Inference Algorithm for Energy Disaggregation

2.1 Introduction

The goal of non-intrusive home appliance load monitoring is to estimate the energy consumption of each appliance in a household, based on the total energy-consumption signal of the household. Such a detailed feedback can help customers in residential areas to save significantly in their electricity bill by changing their consumption behaviors. According to a study by the American Council for an Energy Efficient Economy (ACEEE), providing feedback to customers about their electricity usage patterns may result in an energy savings of up to 19% in residential areas.¹

The bulk of the research in NILM has mostly concentrated on detecting meaningful changes (a.k.a. events) in total power measurements, and associating them with state changes of appliances (e.g., being on or off) by applying different data mining and pattern recognition methods. Several methods, such as artificial neural networks (Chang et al., 2012; Liang et al., 2010), deep neural networks (Kelly & Knottenbelt, 2015a), fuzzy pattern recognition (Lin & Tsai, 2012), nearest neighbor techniques (Chahine et al., 2011; Figueiredo et al., 2012; Weiss et al., 2012), support vector machines and naive Bayes estimation (Chahine et al., 2011), sparse coding (Kolter et al., 2010), and heuristic methods (Dong et al., 2012) have been employed for this purpose.

¹<http://aceee.org/portal/behavior>

Another line of research also takes into account the temporal structure of the data beside the electrical events (Egarter et al., 2015; Guo et al., 2015; Kim et al., 2011; Kolter & Jaakkola, 2012; Parson et al., 2012; Zhong et al., 2014; Zia et al., 2011). Hidden Markov models (HMMs) were used in (Zia et al., 2011) for the NILM problem, where the state evolution of the appliances are modeled by some Markovian dynamics and the total power consumption depends on the joint state of the appliances. A problem with this model is that the number of joint states grows exponentially in the number of appliances, rendering inference computationally infeasible for any realistic scenario. To alleviate this problem, Kim et al. (2011) applied a factorial HMM (FHMM) and its extensions (the most complex being conditional factorial hidden semi-Markov model) to describe the total energy consumption: here each appliance is modeled by an HMM and the total power consumption is the sum of the individual loads (on the negative side, in this work each model has only two states, on and off, which limits the capabilities of the methods).

Although simpler than in the equivalent HMM models, doing exact inference in FHMMs (introduced by Ghahramani & Jordan (1997)) is NP-hard; therefore, computationally efficient approximate methods have been the subject of study. Standard approaches include sampling methods, such as MCMC or particle filtering (Koller & Friedman, 2009) and variational Bayes methods (Ghahramani & Jordan, 1997; Wainwright & Jordan, 2007). In practice, both methods are nontrivial to make work and we are not aware of any works that would have demonstrated good results in our application domain with the type of FHMMs we need to work and at practical scales, including the aforementioned papers (Kim et al., 2011) (using Gibbs sampling) and (Egarter et al., 2015) (applying particle filters). To alleviate this problem, approximate solutions to the inference problem have been proposed which also employ additional information: most notably, Kolter & Jaakkola (2012) achieve state-of-the-art performance by optimizing an objective that is a combination of the log-likelihood coming from the FHMM model and a term capturing the electrical events in the data; they cast their problem as an integer quadratic optimization problem and obtain approximate solution using relaxations leading to a quadratic program. Another interesting approach is that of Zhong et al. (2014), who incorporate domain knowledge by requiring

that the estimated values be close to some specific aggregate values, and approximate the resulting constrained FHMM inference problem by a quadratic program. However, the application of this method is limited by the availability of the target aggregate values (which could be obtained, e.g., from detailed national energy surveys), and even with optimal constraints, the method is inferior to that of Kolter & Jaakkola (2012) (as will be demonstrated by our experiments).

In this chapter we follow the approach of Kolter & Jaakkola (2012) to model the NILM problem by FHMMs and considering the electrical events, as well. The distinguishing features of FHMMs in this setting are that (i) the output is the sum of the output of the underlying HMMs (perhaps with some noise), and (ii) the number of transitions are small compared to the signal length. FHMMs with the first property are called *additive*. In this chapter we derive an efficient, convex relaxation based method for FHMMs of the above type, which significantly outperforms the state-of-the-art algorithms. Our approach is based on revisiting relaxations to the integer programming formulation of Kolter & Jaakkola (2012). In particular, we replace the quadratic programming relaxation of (Kolter & Jaakkola, 2012) with a relaxation to a semi-definite program (SDP), which, based on the literature of relaxations is expected to be tighter and thus better. While SDPs are convex and could in theory be solved using interior-point methods (IPMs) in polynomial time (Malick et al., 2009), IPMs scale poorly with the size of the problem and are thus unsuitable to our large scale problem which may involve as many as a million variables. To address this problem, capitalizing on the structure of our relaxation coming from our FHMM model, we develop a novel variant of the alternating direction method of multipliers (ADMM) (Boyd et al., 2011) that uses *Moreau-Yosida* regularization and combine it with a version of randomized rounding that is inspired by the the recent work of Park & Boyd (2015).

In order to be able to apply the aforementioned FHMM inference methods, we need to set up the FHMM model. This is done by first registering the appliances: that is, collecting some data for each appliance and modeling the resulting power-consumption sequence by an HMM. Typically, expectation maximization (EM) is used to find maximum likelihood estimates of the parameters of an HMM. However, despite its popularity, EM suffers from high computational costs and possibly converging to suboptimal

local optima (Mattfeld, 2014). Therefore, in this chapter we apply (consistent) spectral (or moment) methods to estimate the parameters of HMMs, which are guaranteed to converge to the true parameter values. In particular, we combine the method of Anandkumar et al. (2012) for estimating the output distributions of the HMMs with the algorithm of Kontorovich et al. (2013) to estimate the state transition probability matrix. It will be demonstrated that the proposed combination improves the learning performance significantly.

The rest of the chapter is organized as follows. After defining the notation in Section 2.1.1, we introduce the system model in Section 2.2. SDP relaxations are introduced in Section 2.3 and are applied to the NILM problem in Section 2.4. The main contribution of the chapter, our efficient algorithm to solve the resulting large-scale structured SDP problems, is given in Section 2.4.1. Estimating the HMM parameters is discussed in Section 2.5, while unmodeled appliances are dealt with in Section 2.6. We close the chapter by demonstrating the performance of our algorithms on both synthetic and real world datasets in Section 2.7.

2.1.1 Notation

Throughout the chapter, we use the following notation: $\mathbb{R}$ denotes the set of real numbers, $\mathbb{S}_+^n$ denotes the set of $n \times n$ positive semidefinite matrices, $\mathbb{I}_{\{E\}}$ denotes the indicator function of an event E (that is, it is 1 if the event is true and zero otherwise), $\mathbf{1}$ denotes a vector of appropriate dimension whose entries are all 1. For an integer K , $[K]$ denotes the set $\{1, 2, \dots, K\}$. $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and covariance matrix σ^2 . For a matrix A , we denote its trace by $\mathbf{trace}(A)$, its Frobenius norm by $\|A\|_F$, the vector obtained by concatenating its columns by $\mathbf{vec}(A)$, and the vector formed by its diagonal entries by $\mathbf{diag}(A)$. For two matrices $A, B \in \mathbb{R}^{n \times n}$, $A \succeq B$ denotes that $A - B \in \mathbb{S}_+^n$. For a random variable w , $\mathbb{E}(w)$ denotes its expectation.

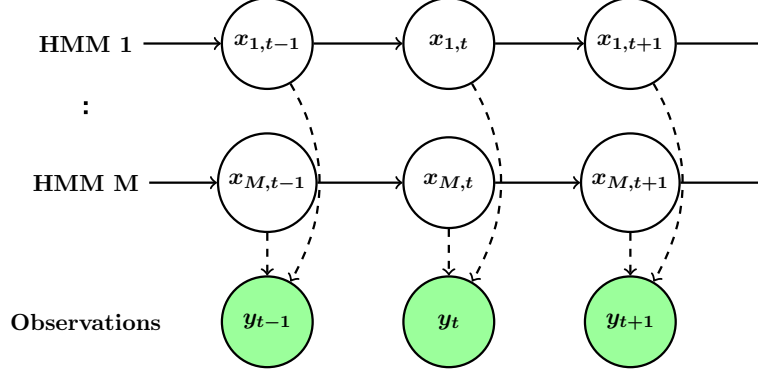


FIGURE 2.1: Graphical representation of FHMMs.

2.2 System Model

Following Kolter & Jaakkola (2012), the energy usage of the household is modeled using an additive factorial HMM (Ghahramani & Jordan, 1997). Suppose there are M appliances in a household. Each of them is modeled via an HMM: let $P_i \in \mathbb{R}^{K_i \times K_i}$ denote the transition-probability matrix of appliance $i \in [M]$, and assume that for each state $s \in [K_i]$, the energy consumption of the appliance is constant $\mu_{i,s}$ (μ_i denotes the corresponding K_i -dimensional column vector $(\mu_{i,1}, \dots, \mu_{i,K_i})^\top$). Denoting by $x_{t,i} \in \{0,1\}^{K_i}$ the indicator vector of the state $s_{t,i}$ of appliance i at time t (i.e., $x_{t,i,s} = \mathbb{I}_{\{s_{t,i}=s\}}$), the total power consumption at time t is $\sum_{i \in [M]} \mu_i^\top x_{t,i}$, and we assume that it is observed with some additive zero mean Gaussian noise of variance σ^2 , that is, $y_t \sim \mathcal{N}(\sum_{i \in [M]} \mu_i^\top x_{t,i}, \sigma^2)$. Fig. 2.1 shows a graphical representation of additive FHMMs, where each Markov chain represents an appliance, while Fig. 2.2 illustrates the model on a real-world signal.

Given this model, the maximum likelihood estimate of the appliance state-vector sequence can be obtained by minimizing the negative log-posterior function

$$\begin{aligned} \underset{x_{t,i}}{\operatorname{argmin}} \quad & \sum_{t=1}^T \frac{(y_t - \sum_{i=1}^M x_{t,i}^\top \mu_i)^2}{2\sigma^2} - \sum_{t=1}^{T-1} \sum_{i=1}^M x_{t,i}^\top (\log P_i) x_{t+1,i} \\ \text{subject to} \quad & x_{t,i} \in \{0,1\}^{K_i}, \mathbf{1}^\top x_{t,i} = 1, i \in [M] \text{ and } t \in [T], \end{aligned} \quad (2.1)$$

where $\log P_i$ denotes a matrix obtained from P_i by taking the logarithm of each entry.

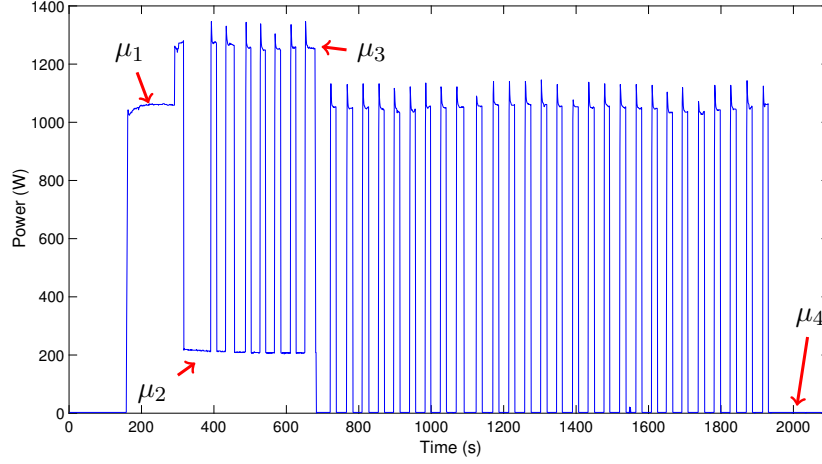


FIGURE 2.2: Sample power consumption of a stove from the REDD dataset (Kolter & Johnson, 2011). Each state i of the appliance is associated with an approximately constant power level μ_i ; in this case $\mu = [1050, 220, 1270, 0]^\top$. The power consumption at time t is modeled as $y_t = x_t^\top \mu + n_t$ where x_t is the state indicator vector (e.g., $x_{350} = [0, 1, 0, 0]^\top$) and n_t is the noise at time t .

In our particular application, in addition to the signal's temporal structure, large changes in total power (in comparison to signal noise) contain valuable information that can be used to further improve the inference results (in fact, solely this information was used for energy disaggregation, e.g., by (Dong et al., 2012; Figueiredo et al., 2012; Weiss et al., 2012)). This observation was used by Kolter & Jaakkola (2012) to amend the posterior with a term that tries to match the large signal changes to the possible changes in the power level when only the state of a single appliance changes.

Formally, let $\Delta y_t = y_{t+1} - y_t$, $\Delta \mu_{m,k}^{(i)} = \mu_{i,k} - \mu_{i,m}$, and define the matrices $E_{t,i} \in \mathbb{R}^{K_i \times K_i}$ by $(E_{t,i})_{m,k} = (\Delta y_t - \Delta \mu_{m,k}^{(i)})^2 / (2\sigma_{\text{diff}}^2)$, for some constant $\sigma_{\text{diff}} > 0$. Intuitively, $(E_{t,i})_{m,k}$ is the negative log-likelihood (up to a constant) of observing a change Δy_t in the power level when appliance i transitions from state m to state k under some zero-mean Gaussian noise with variance σ_{diff}^2 . Making the heuristic approximation that the observation noise and this noise are independent, Kolter & Jaakkola (2012) added the term $(-\sum_{t=1}^{T-1} \sum_{i=1}^M x_{t,i}^\top E_{t,i} x_{t+1,i})$ to the objective of (2.1), arriving to the following

modified version:

$$\underset{x_{t,i}}{\operatorname{argmin}} \left\{ f(x_1, \dots, x_T) := \sum_{t=1}^T \frac{(y_t - \sum_{i=1}^M x_{t,i}^\top \mu_i)^2}{2\sigma^2} - \sum_{t=1}^{T-1} \sum_{i=1}^M x_{t,i}^\top (E_{t,i} + \log P_i) x_{t+1,i} \right\} \quad (2.2)$$

subject to same constraints as in (2.1).

In the next two sections we derive an efficient approximate solution to (2.2); we will demonstrate through extensive experiments in Section 2.7 that it is superior to the approximate solution derived by Kolter & Jaakkola (2012) with respect to several measures quantifying the accuracy of load disaggregation solutions.

2.3 SDP Relaxation and Randomized Rounding

There are two major challenges to solve the optimization problem (2.2) exactly: (i) the objective function is not necessarily convex (namely, the second term); and (ii) the optimization is over binary vectors $x_{t,i}$. To address these problems, we will relax (2.2) to make it an integer quadratic programming problem, then apply an SDP relaxation and randomized rounding to solve approximately the relaxed problem. We start with reviewing the latter methods.

2.3.1 Approximate Solutions for Integer Quadratic Programming

In this section, we consider approximate solutions to the integer quadratic programming problem

$$\begin{aligned} & \text{minimize} && f(x) = x^\top D x + 2d^\top x \\ & \text{subject to} && x \in \{0, 1\}^n, \end{aligned} \quad (2.3)$$

where x is the optimization variable, $D \in \mathbb{S}_+^n$ is positive semidefinite, and $d \in \mathbb{R}^n$.

The standard way to find an approximate optimum of (2.3) is to linearize it by introducing a new variable $X = xx^\top \in \mathbb{S}_+^n$ (then $x^\top D x = \mathbf{trace}(DX)$), and relaxing

the nonconvex constraint $X = xx^\top$ and the integrality constraints $x \in \{0, 1\}^n$ to $X \succeq xx^\top$, $\mathbf{diag}(X) = x$, and $x \in [0, 1]^n$. This leads to the relaxed SDP problem

$$\begin{aligned} & \text{minimize} && \mathbf{trace}(D^\top X) + 2d^\top x \\ & \text{subject to} && \begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \succeq 0, \mathbf{diag}(X) = x, x \in [0, 1]^n, \end{aligned} \quad (2.4)$$

which can be written in the compact SDP form

$$\begin{aligned} & \text{minimize} && \mathbf{trace}(\hat{D}^\top \hat{X}) \\ & \text{subject to} && \hat{X} \succeq 0, \mathcal{A}\hat{X} = b, x \in [0, 1]^n, \end{aligned} \quad (2.5)$$

for appropriate positive semidefinite matrices $\hat{D} = \begin{bmatrix} 0 & d^\top \\ d & D \end{bmatrix}$, $\hat{X} = \begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix} \in \mathbb{S}_+^{n+1}$, where $\mathcal{A} : \mathbb{S}_+^n \rightarrow \mathbb{R}^m$ is a linear operator, and $b \in \mathbb{R}^m$. This general SDP optimization problem can be solved with arbitrary precision in polynomial time using interior-point methods (Malick et al., 2009; Wen et al., 2010); although this approach becomes impractical in terms of both the running time and the required memory if either the number of variables or the optimization constraints are large (Wen et al., 2010). We return to this problem in Section 2.4.1 and Section 2.7.

To obtain a feasible point of (2.3) from the solution of (2.5), we still need to change the solution x to a binary vector. This can be done via randomized rounding (Goemans & Williamson, 1995; Park & Boyd, 2015): Instead of letting $x \in [0, 1]^n$, the integrality constraint $x \in \{0, 1\}^n$ in (2.3) can be replaced by the inequalities $x_i(x_i - 1) \geq 0$ for all $i \in [n]$. Although these lead to nonconvex constraints, they admit an interesting probabilistic interpretation: assuming $w \in \mathbb{R}^n$ is a normal random variable $w \sim \mathcal{N}(\mu, \Sigma)$, the optimization problem (over μ, Σ)

$$\begin{aligned} & \text{minimize} && \mathbb{E}(w^\top Dw + 2d^\top w) \\ & \text{subject to} && \mathbb{E}(w_i(w_i - 1)) \geq 0, \quad i \in [n], \end{aligned}$$

is equivalent to

$$\begin{aligned} & \text{minimize} && \text{trace}((\Sigma + \mu\mu^\top)D) + 2d^\top \mu \\ & \text{subject to} && \Sigma_{i,i} + \mu_i^2 - \mu_i \geq 0, \quad i \in [n], \end{aligned} \quad (2.6)$$

which is in the form of (2.4) with $X = \Sigma + \mu\mu^\top$ and $x = \mu$. This leads to the rounding procedure: starting from a solution (x^*, X^*) of (2.4), we randomly draw several samples $w^{(j)}$ from $\mathcal{N}(x^*, X^* - x^*x^{*\top})$, round $w_i^{(j)}$ to 0 or 1 to obtain $x^{(j)}$, and keep the $x^{(j)}$ with the smallest objective value. This procedure was shown to have better performance experimentally by Park & Boyd (2015) than just naively rounding the coordinates of x^* .

2.4 An Efficient Algorithm for Inference in FHMMs

Now we are ready to apply the results described in the previous section to (2.2). To do so, as mentioned at the beginning of Section 2.3, we need to change the problem to a convex one, since the elements of the second term in the objective of (2.2), $-x_{t,i}^\top(E_{t,i} + \log P_i)x_{t+1,i}$, are not convex. To address this issue, we relax the problem by introducing new variables $Z_{t,i} = x_{t,i}x_{t+1,i}^\top$, and replacing the constraint $Z_{t,i} = x_{t,i}x_{t+1,i}^\top$ with two new ones: $Z_{t,i}\mathbf{1} = x_{t,i}$ and $Z_{t,i}^\top\mathbf{1} = x_{t+1,i}$. To simplify the presentation, we will assume that $K_i = K$ for all $i \in [M]$. Then problem (2.2) becomes

$$\begin{aligned} & \underset{x_{t,i}}{\text{argmin}} && \sum_{t=1}^T \left\{ \frac{1}{2\sigma^2} \left(y_t - x_t^\top \mu \right)^2 - p_t^\top z_t \right\} \\ & \text{subject to} && x_t \in \{0, 1\}^{MK}, \quad t \in [T] \\ & && z_t \in \{0, 1\}^{MKK}, \quad t \in [T-1] \\ & && \mathbf{1}^\top x_{t,i} = 1, \quad t \in [T] \text{ and } i \in [M] \\ & && Z_{t,i}\mathbf{1}^\top = x_{t,i}, \quad t \in [T-1] \text{ and } i \in [M] \\ & && Z_{t,i}^\top\mathbf{1}^\top = x_{t+1,i}, \quad t \in [T-1] \text{ and } i \in [M] \end{aligned} \quad (2.7)$$

where $x_t^\top = [x_{t,1}^\top, \dots, x_{t,M}^\top]$, $\mu^\top = [\mu_1^\top, \dots, \mu_M^\top]$, $z_t^\top = [\mathbf{vec}(Z_{t,1})^\top, \dots, \mathbf{vec}(Z_{t,M})^\top]$, and $p_t^\top = [\mathbf{vec}(E_{t,1} + \log P_1)^\top, \dots, \mathbf{vec}(E_{t,M} + \log P_M)^\top]$.

Expanding the first term of (2.7) and following the relaxation method of Section 2.3.1, we can easily get the following SDP problem:²

$$\begin{aligned} \underset{X_t, z_t}{\operatorname{argmin}} \quad & \sum_{t=1}^T \operatorname{trace}(D_t^\top X_t) + d_t^\top z_t \\ \text{subject to} \quad & \mathcal{A}X_t = b, \quad \mathcal{B}X_t + \mathcal{C}z_t + \mathcal{E}X_{t+1} = g \\ & X_t \succeq 0, \quad X_t, z_t \geq 0 \end{aligned} \tag{2.8}$$

where $\mathcal{A} : \mathbb{S}_+^{MK+1} \rightarrow \mathbb{R}^m$, $\mathcal{B}$ and $\mathcal{E} : \mathbb{S}_+^{MK+1} \rightarrow \mathbb{R}^{m'}$, and $\mathcal{C} \in \mathbb{R}^{MKK \times m'}$ are all linear operators, and the integers m and m' are determined by the number of equality constraints. Here $D_t = \frac{1}{2\sigma^2} \begin{bmatrix} 0 & -y_t \mu^\top \\ -y_t \mu & \mu \mu^\top \end{bmatrix}$ and $d_t = p_t$. Notice that (2.8) is a simple, but large-scale SDP problem in the form of (2.5) where $\hat{D}$ has a special block structure. In the following, we will utilize this structure to provide an efficient approximate solution based on ADMM, and use randomized rounding, as described in Section 2.3.1, to find a good integer solution.

2.4.1 Scalable Solutions for Large SDP Problems

Although interior-point methods can solve SDP problems efficiently, even for problems with sparse constraints as (2.4), the running time to obtain an ϵ optimal solution for n variables is of the order of $n^{3.5} \log(1/\epsilon)$ (Nesterov, 2004, Section 4.3.3), which becomes prohibitive in our case since the number of variables scales linearly with the time horizon T . In fact, for one day of data sampled at 1 Hz yields $T = 86400$ seconds, and with $M = 10$ appliances with $K = 4$ states each, the number of variables becomes $MKT \approx 3,500,000$, rendering the application of interior-point methods clearly infeasible.

²The only modification is that we need to keep the equality constraints in (2.7) that are missing from (2.3).

As an alternative solution, *first-order* methods can be used for large scale problems. In particular, we are interested in the alternating direction method of multipliers (ADMM).

2.4.1.1 Background: Alternating Direction Method of Multipliers (ADMM)

In this section, we introduce ADMM based on the paper of Boyd et al. (2011). ADMM is an algorithm for solving large-scale optimization problems by breaking them into smaller subproblems and it is well suited to many decentralized and “big data” applications that arise in machine learning, statistics and other modern applications. The method uses a procedure called *decomposition-coordination*, in which the solutions of some smaller problems are coordinated in a way that they achieve a global solution of a large problem. ADMM was originally developed in the 1970s and since then it has been re-invented in different fields. Douglas-Rachford splitting from numerical analysis³, Dykstra’s alternating projections method and Bregman iterative algorithms for l_1 problems in signal processing are example of algorithms from different fields that are either equivalent or closely related to ADMM. The method has been recently rediscovered due to advancement in distributed computing to solve massive optimization problems.

In its simplest form, ADMM solves problems in the following form:

$$\begin{aligned} & \text{minimize} && f(x) + g(z) \\ & \text{subject to} && \mathcal{A}x + \mathcal{B}z = c. \end{aligned} \tag{2.9}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^m \rightarrow \mathbb{R}$ are convex functions, $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ are the optimization variables, $\mathcal{A} \in \mathbb{R}^{p \times n}$, $\mathcal{B} \in \mathbb{R}^{p \times m}$ and $c \in \mathbb{R}^p$ for some dimensions $n, m, p > 0$. For problem (2.9), the *augmented Lagrangian* function can be defined as

$$L_\rho(x, z, y) = f(x) + g(z) + y^\top (\mathcal{A}x + \mathcal{B}z - c) + \frac{\rho}{2} \|\mathcal{A}x + \mathcal{B}z - c\|_2^2,$$

³Douglas-Rachford splitting was developed in the 1950s for the numerical solution of partial differential equations (Parikh & Boyd, 2013).

where $\rho > 0$ and $y \in \mathbb{R}^p$. Then, ADMM becomes a procedure that goes through the following steps iteratively to find the optimal solution:

$$x^{k+1} \triangleq \underset{x}{\operatorname{argmin}} L_\rho(x, z^k, y^k), \quad (2.10)$$

$$z^{k+1} \triangleq \underset{z}{\operatorname{argmin}} L_\rho(x^{k+1}, z, y^k), \quad (2.11)$$

$$y^{k+1} \triangleq y^k + \rho \left(\mathcal{A}x^{k+1} + \mathcal{B}z^{k+1} - c \right), \quad (2.12)$$

where k is the iteration counter. It can be seen that ADMM reduces the optimization problem (2.9) to an x -minimization step (2.10), a z -minimization step (2.11) and a dual variable update (2.12), and handles each step separately. Indeed, handling the objective terms independently of each other is the main advantage of ADMM, which helps with optimization problems where f and g can be optimized efficiently, but optimizing their sum directly (given the joint constraints on the variables) is not possible or is hard (Parikh & Boyd, 2013).

2.4.1.2 ADMM for SDP Problems

Since our problem (2.8) is an SDP problem where the objective function is separable, ADMM is a promising candidate to find a near-optimal solution. Utilizing its sparse structure, we can consider the sequence of SDP problems over the variables (X_t, z_t) for $t = 1, \dots, T$ as

$$\begin{aligned} & \underset{X_t, z_t}{\operatorname{argmin}} && \mathbf{trace}(D_t^\top X_t) + d_t^\top z_t \\ & \text{subject to} && \mathcal{A}X_t = b, X_t \succeq 0, X_t, z_t \geq 0 \\ & && \mathcal{B}X_t + \mathcal{C}z_t + \mathcal{E}X_{t+1} = g \\ & && \mathcal{B}X_{t-1} + \mathcal{C}z_{t-1} + \mathcal{E}X_t = g \end{aligned} \quad (2.13)$$

(X_{T+1} and z_T can be chosen arbitrarily). Note that the optimization problems are connected via the linear constraints: optimizing for t depends on the variables for $t-1$ and $t+1$. Applying the idea of ADMM, we update the variables in repeated passes for $t = 1, \dots, T$, using *Moreau-Yosida* quadratic regularization (Malick et al., 2009), suited for the primal formulation we consider: The regularized Lagrangian function for

(2.13) is⁴

$$\begin{aligned}
\mathcal{L}_\mu = & \text{trace}(D^\top X) + d^\top z + \frac{1}{2\mu} \|X - S\|_F^2 + \frac{1}{2\mu} \|z - r\|_2^2 \\
& + \lambda^\top (b - \mathcal{A}X) + \nu^\top (g - \mathcal{B}X - \mathcal{C}z - \mathcal{E}X_+) \\
& + \nu_-^\top (g - \mathcal{B}X_- - \mathcal{C}z_- - \mathcal{E}X) \\
& - \text{trace}(W^\top X) - \text{trace}(P^\top X) - h^\top z,
\end{aligned} \tag{2.14}$$

where $\lambda, \nu, W \geq 0$, $P \succeq 0$, and $h \geq 0$ are dual variables, S and r are regularizers where S is symmetric, and $\mu > 0$ is a constant step size parameter. One can readily see that $\mathcal{L}_\mu$ is a function of both primal and dual variables; so we first minimize $\mathcal{L}_\mu$ in X and z to get $\hat{\mathcal{L}}_\mu$ which is a function of only dual variables. Then, the standard ADMM updates (Malick et al., 2009) are obtained by sequentially updating the dual variables to minimize $\hat{\mathcal{L}}_\mu$. Finding the minima by setting the corresponding derivatives to zero, we obtain the following updates:

$$\begin{aligned}
\nabla_X \mathcal{L}_\mu &= D + \frac{1}{\mu}(X - S) - \mathcal{A}^\top \lambda - \mathcal{B}^\top \nu - \mathcal{E}^\top \nu_- - W - P = 0, \\
X^* &= S + \mu(\mathcal{A}^\top \lambda + \mathcal{B}^\top \nu + \mathcal{E}^\top \nu_- + W + P - D)
\end{aligned}$$

and

$$\begin{aligned}
\nabla_z \mathcal{L}_\mu &= d + \frac{1}{\mu}(z - r) - \mathcal{C}^\top \nu - h = 0, \\
z^* &= r + \mu(\mathcal{C}^\top \nu + h - d).
\end{aligned}$$

Substituting $X = X^*$ and $z = z^*$ in (2.14) defines $\hat{\mathcal{L}}_\mu = \min_{X,z} \mathcal{L}_\mu$. Then the standard ADMM updates become

$$\begin{aligned}
W^{k+1} &= \underset{W \geq 0}{\operatorname{argmin}} \hat{\mathcal{L}}_\mu(S^k, P^k, W, \lambda^k, \nu^k, \nu_-^k, h^k, r^k) \\
&= \max \{ (D - \mathcal{A}^\top \lambda^k - \mathcal{B}^\top \nu^k - \mathcal{E}^\top \nu_-^k - P^k - S^k/\mu), \mathbf{0} \}, \\
P^{k+1} &= \underset{P \succeq 0}{\operatorname{argmin}} \hat{\mathcal{L}}_\mu(S^k, P, W^{k+1}, \lambda^k, \nu^k, \nu_-^k, h^k, r^k) \\
&= (D - \mathcal{A}^\top \lambda^k - \mathcal{B}^\top \nu^k - \mathcal{E}^\top \nu_-^k - W^{k+1} - S^k/\mu)_+, \\
\lambda^{k+1} &= \underset{\lambda}{\operatorname{argmin}} \hat{\mathcal{L}}_\mu(S^k, P^{k+1}, W^{k+1}, \lambda, \nu^k, \nu_-^k, h^k, r^k)
\end{aligned}$$

⁴We drop the subscript t and replace $t \pm 1$ with $+$ and $-$ signs, respectively.

$$\begin{aligned}
&= \frac{1}{\mu} (\mathcal{A}\mathcal{A}^\top)^\dagger \left(b - \mathcal{A}(\mathcal{B}^\top \nu^k + \mathcal{E}^\top \nu_-^k + W^{k+1} + P^{k+1} - D) \right), \\
S^{k+1} &= S^k + \mu (\mathcal{A}^\top \lambda^{k+1} + \mathcal{B}^\top \nu^k + \mathcal{E}^\top \nu_-^{k+1} + W^{k+1} + P^{k+1} - D), \\
h^{k+1} &= \underset{h \geq 0}{\operatorname{argmin}} \hat{\mathcal{L}}_\mu(S^{k+1}, P^{k+1}, W^{k+1}, \lambda, \nu^k, \nu_-^k, h, r^k) \\
&= \max \left\{ (d - \mathcal{C}^\top \nu^k - r^k / \mu), \mathbf{0} \right\}, \\
\nu^{k+1} &= \underset{\nu}{\operatorname{argmin}} \hat{\mathcal{L}}_\mu(S^{k+1}, P^{k+1}, W^{k+1}, \lambda^{k+1}, \nu, \nu_-^{k+1}, h^{k+1}, r^k) \\
&= \frac{1}{\mu} (\mathcal{B}\mathcal{B}^\top + \mathcal{C}\mathcal{C}^\top + \mathcal{E}\mathcal{E}^\top)^\dagger \\
&\quad \times \left(g - \mathcal{B}(S^{k+1} + \mu(\mathcal{A}^\top \lambda^{k+1} + \mathcal{E}^\top \nu_-^{k+1} + W^{k+1} \right. \\
&\quad \quad \left. + P^{k+1} - D)) - \mathcal{C}(r^k + \mu(h^{k+1} - d)) \right. \\
&\quad \quad \left. - \mathcal{E}(S_+^k + \mu(\mathcal{A}^\top \lambda_+^k + \mathcal{B}^\top \nu_+^k + W_+^k + P_+^k - D)) \right), \\
r^{k+1} &= r^k + \mu(\mathcal{C}^\top \nu^{k+1} + h^{k+1} - d),
\end{aligned} \tag{2.15}$$

where $\max : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{X}$ works elementwise, and for any square matrix A , $A^\dagger$ denotes the *Moore-Penrose pseudo inverse*, and for any real symmetric matrix A , A_+ is the projection of A onto the positive semidefinite cone (given the spectral decomposition $A = \sum_i \lambda_i v_i v_i^\top$, where λ_i , and v_i are the eigenvalues and, respectively, corresponding eigenvectors of A , $A_+ = \sum_{\lambda_i > 0} \lambda_i v_i v_i^\top$).

The final algorithm, which updates the variables for each t sequentially, is given in Algorithm 1. Analyzing the exact form of the updates, it is clear that—due to the computation of some pseudo inverses—a single iteration in the algorithm is of $O((MK)^3 T)$ computational complexity.

Algorithm 1 ADMM for sparse SDPs of the form (2.8).

Input: length of input data: T , number of iterations: `itermax`.

Initialization: Set the initial values to zero: $W_t^0, P_t^0, S^0 = \mathbf{0}$, $\lambda_t^0 = \mathbf{0}$, $\nu_t^0 = \mathbf{0}$, and $r_t^0, h_t^0 = \mathbf{0}$. Set μ to some small value (default used is 0.001).

for $k = 0, 1, \dots, \text{itermax}$ **do**

Update W_t^k , P_t^k , λ_t^k , S_t^k , r_t^k , h_t^k , and ν_t^k , respectively, sequentially for all $t = 1, 2, \dots, T$, according to (2.15).

end for

Algorithm 2 Randomized rounding to find an approximate solution of (2.2).

Input: number of iterations: itermax, length of input data: T .

Initialization: Solve (2.8) (e.g., solve it approximately via Algorithm 1) to get X_t^* and z_t^* and set $x_t^{best} := z_t^*$ and $X_t^{best} := X_t^*$ for all $t \in [T]$.

for $t = 2, \dots, T - 1$ **do**

Set $x := [x_{t-1}^{best\top}, x_t^{best\top}, x_{t+1}^{best\top}]^\top$.

Set $X := \mathbf{block}(X_{t-1}^{best}, X_t^{best}, X_{t+1}^{best})$ where **block** constructs a block-diagonal matrix from its arguments.

Set $f^{best} := \infty$.

Form covariance matrix $\Sigma := X - xx^T$, and find its Cholesky factorization $CC^\top = \Sigma$.

for $k = 1, 2, \dots, \text{itermax}$ **do**

Random sampling: $z^k := x + Cw$, where $w \sim \mathcal{N}(0, I)$.

Round z^k to the nearest integer point x^k that satisfies the constraints in (2.7).

If $f^{best} > f_t(x^k)$ then update best point x_t^{best} and X_t^{best} from the corresponding entries of x^k and $x^k x^{k\top}$, respectively.

end for
end for

2.4.2 Randomized Rounding

Starting from a (possibly approximate) solution (z^*, X^*) of (2.8), we apply the randomized rounding method from Section 2.3.1 to provide an approximate solution to our original problem (2.2). Utilizing that we have an SDP problem for each time step $t \in [T]$, Algorithm 2 performs the rounding sequentially for every t . Note that due to the dependence on time steps $t \pm 1$ in (2.13) and (2.7), rounding is performed over triplets of variables $(x_{t-1}^{best}, x_t^{best}, x_{t+1}^{best})$, although we just keep the updated values for time t . Note that the most expensive step in the algorithm is to find the Cholesky factorization of Σ , which has $O(M^3 K^3)$ computational complexity.

Following Park & Boyd (2015), a simple greedy local search is introduced within Algorithm 2 in the experiments. After finding the point x^k , we try to improve the objective by changing the state of individual appliances greedily: we go over the appliances sequentially (starting with the first one), check all possible states of the actual appliance and keep the one with the minimum objective value (keeping the original state if no improvement is possible). Here we use another heuristic and encourage sparse solutions in the sense that we try to explain the observed signal with as few appliances as possible. Therefore, we augment the objective in (2.7) by a penalty term corresponding to the

number of active appliances, and minimize $f_t(x^k) = \frac{1}{2\sigma^2}(y_t - x_t^k \mu)^2 - p_t^\top z_t + L\|\bar{x}_t^k\|_1$ where x_t^k is the middle component of x_k , $\bar{x}_t^k$ is the indicator vector of the on states only (i.e., obtained by removing the components of x_t^k corresponding to off states) and $L > 0$ is a suitable constant. The process is repeated as long as the objective can be improved this way.

Combining Algorithms 1 and 2 provides an efficient method for finding an approximate solution of (2.2), thus solving the additive FHMM inference problem. The overall computational complexity of each iteration of the combined algorithm is $O((MK)^3T)$, which is a significant reduction from the $O((MKT)^{3.5})$ complexity of the interior point methods for any practical scenario ($T \gg MK$, see Section 2.4.1), even if the number of iterations is on the order of a few thousands.

2.5 Learning the Model

The previous section provided an algorithm to solve the inference part of our energy disaggregation problem. However, to be able to run the inference method, we need to set up the model. Most often the parameters of an HMM are estimated from data by the EM algorithm, which is a heuristic local search method for maximum likelihood parameter estimation. The problem with the EM algorithm is that it converges slowly and is prone to local optima. Therefore, alternative solutions have been proposed in the literature (Anandkumar et al., 2012; Cybenko & Crespi, 2011; Hsu et al., 2012; Kontorovich et al., 2013; Siddiqi et al., 2010). In particular, *spectral learning* techniques (e.g., Anandkumar et al. 2012; Hsu et al. 2012) based on the method of moments (Pearson, 1894) and spectral decompositions (Chang, 1996) have gained popularity in recent years. In addition, the method of Kontorovich et al. (2013)—assuming the conditional distribution of observations given the hidden states is known—formulates the estimation of the transition matrix as a quadratic program (we call it the *QP method* hereafter), which can be solved by using any off-the-shelf quadratic program package. In contrast to EM, these algebraic methods are consistent (i.e., converge to the global optimum) and have more reasonable computational and sample complexity (i.e., the matrix of second-moments and the tensor of third-moments are estimated

only once and through a single pass of the algorithm over the observation data). In what follows, we review these methods.

2.5.1 A Spectral Learning Method for Discrete-Output HMMs

In this section, we summarize the main results of Anandkumar et al. (2012). Consider a discrete HMM with K hidden states and D distinct observation symbols. Let $P \in \mathbb{R}^{K \times K}$ be the transition matrix describing the underlying Markov chain, and let $\pi \in \mathbb{R}^K$ be its initial state distribution. Let $y_t \in \{e_1, \dots, e_D\}$ denote the one-hot encoding of the observation at time t , where $e_i \in \mathbb{R}^D$ is the i^{th} standard unit vector in the Euclidean space: $y_t = e_i$ if at time t symbol i is observed. With a slight abuse of notation, let the tuple (y_1, y_2, y_3) denote three consecutive observations.⁵ Define $O = [o_1 | o_2 | \dots | o_K] \in \mathbb{R}^{D \times K}$ to be the matrix of the conditional means of the observation y_t at time t given the corresponding hidden state s_t , i.e., $\mathbb{E}[y_t | s_t = i] = o_i$ for all $i \in [K]$. The following parameters can be defined:

$$w \triangleq P\pi, \quad M_1 \triangleq O \text{diag}(\pi) P^\top \text{diag}(P\pi)^{-1}, \quad M_2 \triangleq O, \quad \text{and} \quad M_3 \triangleq OP.$$

Define the matrix of second-order moments $C_{3,1} \in \mathbb{R}^{D \times D}$ as

$$(C_{3,1})_{i,j} \triangleq \Pr[y_3 = e_i, y_1 = e_j] \quad \text{for } i, j \in [D],$$

and the tensor of third-order moments $C_{3,1,2}$ as

$$(C_{3,1,2})_{i,j,k} \triangleq \Pr[y_3 = e_i, y_1 = e_j, y_2 = e_k] \quad \text{for } i, j, k \in [D].$$

These definitions allow to view $C_{3,1}$ and $C_{3,1,2}$ as expectations of tensor products of y_1, y_2 and y_3 :

$$C_{3,1} \triangleq \mathbb{E}[y_3 \otimes y_1] \quad \text{and} \quad C_{3,1,2} \triangleq \mathbb{E}[y_3 \otimes y_1 \otimes y_2],$$

⁵Here we focus on (y_1, y_2, y_3) , but the same properties holds for other three consecutive observations (y_{t-1}, y_t, y_{t+1}) for every $t \in \{2, \dots, T-1\}$.

where $\otimes$ denotes tensor product. Also, the linear operator $C_{3,1,2} : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times D}$ is defined as

$$C_{3,1,2}(\eta) \triangleq \mathbb{E}[(y_3 \otimes y_1) \langle \eta, y_2 \rangle],$$

where $\eta \in \mathbb{R}^D$. In other words, the (i, j) -th entry of $C_{3,1,2}(\eta)$ is

$$C_{3,1,2}(\eta)_{i,j} = \sum_{l=1}^D \eta_l (C_{3,1,2})_{i,j,l}.$$

The following lemma shows that $C_{3,1}$ and $C_{3,1,2}(\eta)$ can be seen as certain matrix products which involve the model parameters P , O and π .

Lemma 2.1. *We have $C_{3,1} = M_3 \text{diag}(w) M_1^\top$ and $C_{3,1,2}(\eta) = M_3 \text{diag}(M_2 \eta) \text{diag}(w) M_1^\top$ for $\eta \in \mathbb{R}^D$.*

In addition, Lemma 2.2 defines a matrix which is called “observable operator.” This matrix uses the pairwise and tripletwise probabilities in a way that reveals O .

Lemma 2.2. *For $i \in \{1, 2, 3\}$, let $U_i \in \mathbb{R}^{D \times K}$ be a matrix such that $U_i^\top M_i \in \mathbb{R}^{K \times K}$ is invertible. Then $U_3^\top C_{3,1} U_1$ is invertible, and for all $\eta \in \mathbb{R}^D$, the “observable operator” $B_{3,1,2}(\eta) \in \mathbb{R}^{K \times K}$, defined by $B_{3,1,2}(\eta) \triangleq (U_3^\top C_{3,1,2}(\eta) U_1) (U_3^\top C_{3,1} U_1)^{-1}$, satisfies*

$$B_{3,1,2}(\eta) = \left(U_3^\top M_3 \right) \text{diag} (M_2 \eta) \left(U_3^\top M_3 \right)^{-1}. \quad (2.16)$$

In particular, the K roots of the characteristic polynomial $\lambda \mapsto \det (B_{3,1,2}(\eta) - \lambda I)$ are $\{\langle \eta, M_{2,j} \rangle, j \in [K]\}$, where $M_{2,j}$ is the j^{th} column of M_2 .

By substituting the definition of M_2 and M_3 into (2.16) we have

$$B_{3,1,2}(\eta) = \left(U_3^\top O P \right) \text{diag} (O \eta) \left(U_3^\top O P \right)^{-1}.$$

This form allows us to compute the observable operator matrix O from the diagonalization of $B_{3,1,2}(\eta)$. The following lemma shows how we can recover O .

Lemma 2.3. *Given the conditions and definitions of Lemma 2.2, let $\Theta \in \mathbb{R}^{K \times K}$ be an invertible matrix, and let $\theta_i \in \mathbb{R}^K$ be its i -th row. Moreover, for all $i \in [K]$, let $\lambda_{i,1}, \dots, \lambda_{i,K}$ denote the K eigenvalues of $B_{3,1,2}(U_2 \theta_i)$ in the order specified by the*

Algorithm 3 Spectral learning for HMMs (the moments method) Anandkumar et al. (2012)

- 1: Transform the observation sequence $\{y_t\}_{t=1}^T$ to triplets thorough a sliding window of length three, such that the entries of each tuple are three consecutive samples from the observation sequence.
- 2: Compute empirical estimates for $\hat{C}_{3,1} \in \mathbb{R}^{D \times D}$, $\hat{C}_{3,2} \in \mathbb{R}^{D \times D}$, and $\hat{C}_{3,1,2} \in \mathbb{R}^{D \times D \times D}$ from these triplets as

$$\begin{aligned}
 (\hat{C}_{3,1})_{i,j} &= \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbb{I}_{\{y_{t+2}=e_i\}} \mathbb{I}_{\{y_t=e_j\}} \\
 (\hat{C}_{3,2})_{i,j} &= \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbb{I}_{\{y_{t+2}=e_i\}} \mathbb{I}_{\{y_{t+1}=e_j\}} \\
 (\hat{C}_{3,1,2})_{i,j,k} &= \frac{1}{T-2} \sum_{t=1}^{T-2} \mathbb{I}_{\{y_{t+2}=e_i\}} \mathbb{I}_{\{y_t=e_j\}} \mathbb{I}_{\{y_{t+1}=e_k\}}
 \end{aligned}$$

- 3: Let $\hat{U}_3 \in \mathbb{R}^{D \times K}$ and $\hat{U}_1 \in \mathbb{R}^{D \times K}$ be matrices formed of the orthonormal left and right singular vectors of $\hat{C}_{3,1}$ corresponding to its top K singular values, respectively. Let $\hat{U}_2 \in \mathbb{R}^{D \times K}$ be the matrix of the orthonormal right singular vectors of $\hat{C}_{3,2}$ corresponding to its top K singular values.
 - 4: Pick a random rotation matrix $\Theta \in \mathbb{R}^{K \times K}$, with its i -th row denoted by $\theta_i^T \in \mathbb{R}^K$. Θ . Compute $\hat{B}_{3,1,2}(\hat{U}_2\theta_1) = \left(\hat{U}_3^T \hat{C}_{3,1,2}(\hat{U}_2\theta_1) \hat{U}_1 \right) \left(\hat{U}_3^T \hat{C}_{3,1} \hat{U}_1 \right)^{-1}$.
 - 5: Compute $\hat{R}_3 \in \mathbb{R}^{K \times K}$ that diagonalizes $B_{3,1,2}(\hat{U}_2\theta_1)$, i.e., $\hat{R}_3^{-1} \hat{B}_{3,1,2}(\hat{U}_2\theta_1) \hat{R}_3 = \text{diag}(\hat{\lambda}_{1,1}, \hat{\lambda}_{1,2}, \dots, \hat{\lambda}_{1,K})$.
 - 6: For each $i \in \{2, \dots, K\}$, find the diagonal entries $\hat{\lambda}_{i,1}, \hat{\lambda}_{i,2}, \dots, \hat{\lambda}_{i,K}$ of $\hat{R}_3^{-1} \hat{B}_{3,1,2}(\hat{U}_2\theta_1) \hat{R}_3$, and form the matrix $L \in \mathbb{R}^{K \times K}$ that its (i,j) -th entry is $\hat{\lambda}_{i,j}$.
 - 7: Return $\hat{O} \triangleq \hat{U}_2 \Theta^{-1} \hat{L}$.
-

matrix of right eigenvectors $U_2^\top M_2$. Let $L \in \mathbb{R}^{K \times K}$ be the matrix whose (i,j) -th entry is $\lambda_{i,j}$. Then

$$\Theta U_2^\top M_2 = L. \quad (2.17)$$

By replacing M_2 with its definition ($M_2 \triangleq O$) and multiplying the left-hand side of (2.17) by $U_2 \Theta^{-1}$, we get

$$O = U_2 \Theta^{-1} L.$$

Algorithm 3 shows how to estimate the *conditional mean observation matrix* O from an observation sequence.

2.5.2 The QP Learning Method

In this section we review the QP algorithm of Kontorovich et al. (2013) for learning a parametric output HMM. The algorithm is based on the following insights: (i) the stationary distribution of an *ergodic* HMM is a mixture of distributions from a parametric family; (ii) assuming the conditional distribution of the output given the states is known, the transition probabilities can be computed from the observation signal. Kontorovich et al. (2013) proposed a learning algorithm that works with both continuous- and discrete-output HMMs. The proposed algorithm decouples learning of the parameters of the output distribution from learning the transition probabilities as follows: (i) the parameters of the output distribution are estimated by a “general purpose parametric mixture learner”;⁶ and (ii) given the estimates of the parameters of the output distribution, the transition probabilities can be estimated by solving a quadratic program.

2.5.2.1 Learning Discrete-Output HMM

First we describe the QP algorithm for discrete-output HMMs—the extension to the continuous case is explained in the last section. The following lemma establishes the connection between the stationary distribution of the observations and the stationary distribution of the underlying Markov process.

Lemma 2.4. *Let $\pi \in \mathbb{R}^K$ be the stationary distribution of a Markov chain with transition matrix $P \in \mathbb{R}^{K \times K}$, and let B be the emission matrix, $B_{i,j} \triangleq \Pr[y_t = i | s_t = j]$. Moreover, let ρ be the stationary distribution of the observations $\{y_t\}_{t=1}^T$. Then $\rho = B\pi$.*

Assuming the emission matrix is given, an empirical estimate $\hat{\rho}$ of the stationary distribution of the observations can be computed from a single pass on the data by computing the empirical frequencies of the output symbols. Based on Lemma 2.4, one can compute an empirical estimate $\hat{\pi}$ of π as the solution of the following *quadratic program*:

$$\hat{\pi} = \arg \min_{x_i > 0, \|x\|_1 = 1} \| \hat{\rho} - Bx \|_{(\frac{1}{\hat{\rho}})}^2$$

⁶Learning the output parameters are beyond the scope of the QP learning algorithm.

where $\|x\|_w^2 = \sum_k w_k x_k^2$ is the squared weighted l_2 norm with weight vector w , and $\frac{1}{\hat{\rho}}$ stands for the vector whose i^{th} component is $\frac{1}{\hat{\rho}_i}$.

In order to estimate P , consider the pair (y_t, y_{t+1}) of the consecutive observations. By definition, we have

$$\Pr[y_t = k, y_{t+1} = k'] = \sum_{i,j \in [K]} B_{k',i} B_{k,j} P_{i,j} \Pr[s_t = j]$$

for all $t \in \{1, \dots, T-1\}$ where T is the length of the observation sequence. Given π and B , we can define the matrix of second-order moments, $C_{1,2} \in \mathbb{R}^{D \times D}$, as $C_{1,2} \triangleq \mathbb{E}[y_t \otimes y_{t+1}]$. By definition, the (k, k') -th entry of $C_{1,2}$ is

$$(C_{1,2})_{k,k'} = \sum_{i,j} C_{i,j}^{k,k'} P_{i,j} \quad \text{where} \quad C_{i,j}^{k,k'} = \pi_j B_{k,j} B_{k',i}. \quad (2.18)$$

Notice that $C_{1,2}$ can be estimated based on data. Defining the operator $D_{1,2}(P) : \mathbb{R}^{K \times K} \rightarrow \mathbb{R}^{K \times K}$ by $(D_{1,2}(P))_{k,k'} = \sum_{i,j} C_{i,j}^{k,k'} P_{i,j}$, equation (2.18) can be written as $D_{1,2}(P) = C_{1,2}$. If $\hat{C}_{1,2}$ denotes the empirical estimate of $C_{1,2}$ and $\hat{D}_{1,2}(P)$ denotes the empirical estimate of $D_{1,2}(P)$ calculated by replacing π and B with their empirical estimates, $\hat{\pi}$ and $\hat{B}$, in (2.18), Kontorovich et al. (2013) proposed to estimate P by solving the QP

$$\hat{P} = \arg \min_{P_{i,j} \geq 0, \sum_i P_{i,j} = 1, P\hat{\pi} = \hat{\pi}} \left\| \text{vec}(\hat{C}_{1,2}) - \text{vec}(\hat{D}_{1,2}(P)) \right\|_{\left(\frac{1}{\text{vec}(\hat{C}_{1,2})}\right)}^2$$

2.5.2.2 Learning Continuous-Output HMM

In order to handle continuous-output HMMs, we can map each observation y_t to a K -dimensional vector $\varphi(y_t) = [f_{\theta_1}(y_t), \dots, f_{\theta_K}(y_t)]^T$, where f_{θ_i} denotes a Gaussian density with parameters θ_i . By this trick, we can use the same method as the discrete case. In the continuous case, the “effective observation matrix” K can be defined as

$$K_{i,j} = \int_{\mathcal{Y}} f_{\theta_i}(y) P(y|j) dy,$$

where the distribution $P(y|j)$ can be found by any Gaussian mixture model parameter estimation method (Kontorovich et al., 2013). In this way, the matrix K plays the same role as emission matrix B . So, by replacing B with K , the rest of the algorithm remains unchanged.

2.5.3 Learning HMMs for NILM Application

In this chapter we use the method of Anandkumar et al. (2012) to estimate the underlying HMM model from a sequence of observations coming from a single appliance. While this method is directly applicable to continuous output distributions if the dimensionality of the output is larger than the number of states, this is clearly violated in our case as we have one-dimensional observations. Therefore, during the learning process: (i) the observed data is discretized; (ii) a discrete output distribution for each state is created by applying the learning algorithm; (iii) the discrete output distributions are turned into continuous distributions with piecewise uniform densities; and (iv) the final Gaussian estimate for each output distribution is obtained by minimizing the KL divergence from the corresponding piecewise uniform density.

While the above estimation method can also provide an estimate of the transition matrix, in practice, we found that better performance can be achieved by the estimation method of Kontorovich et al. (2013). In Section 2.7.1 we present experimental results in a controlled environment supporting our claims about the performance of the different algorithms.

2.6 Unmodeled Appliances

When it comes to the specific application to NILM, the problem of *unknown, time-varying bias* also needs to be addressed, which appears due to the presence of unknown/unmodeled appliances in the measured signal. Dealing with this problem is one of the most challenging part of any NILM project, but not much has been done yet. Kolter & Jaakkola (2012) proposed to use a generic model with a smaller weight in the objective function, but incorporating this idea in the FHMM inference imposes

great technical challenges, which comes from the fact that the inference problem is NP-complete by itself, let alone introducing the generic model. Even Kolter & Jaakkola (2012) faced difficulty in linearizing the problem in the presence of this generic model, and had to introduce extra relaxation beyond the assumption that at most one appliance can change its state at any given time.⁷

Therefore, in this work we use a *practical, heuristic* solution for NILM. First, we identify all electrical events defined by a large change Δy_t in the power usage (20W in the experiments). Then, we discard all events that are close to any possible level change $\Delta \mu_{m,k}^{(i)}$ (within $\pm 10\%$ in the experiments). The remaining large jumps are regarded as coming from a generic HMM model describing the unregistered appliances: they are clustered into $K - 1$ clusters, and an HMM model is built where each cluster is regarded as power usage coming from a single state of the unregistered appliances, and we allow an off state with power usage 0.

2.7 Experimental Results

In this section we present experiments demonstrating the capabilities of our algorithm and compare its performance to other state-of-the-art methods⁸.

2.7.1 Learning the Parameters of an HMM

We tested the performance of several methods for parameter estimation in HMMs: EM, the algorithm of Hsu et al. (2012) (Spectral), that of Anandkumar et al. (2012) (Moments), and the combined algorithm of Anandkumar et al. (2012) and Kontorovich et al. (2013) described in Section 2.5 to estimate the transition matrix (QP).⁹ Fig. 2.3 shows the performance of these algorithms in a $K = 3$ state discrete-output HMM, where the output alphabet size is $D = 8$: The emission matrix O ($O_{i,j}$ denotes the probability of observing i in state j) and the transition matrix P are given in the left column, while the right column shows the mean squared-error in estimating the

⁷In fact, the subtle argument showing the need for further relaxation skipped the attention of the authors, see Appendix 2.A for details.

⁸Our code is available online at https://github.com/kiarashshaloudegi/FHMM_inference.

⁹The implementation of the algorithms Spectral and Moments is taken from Mattfeld (2014).

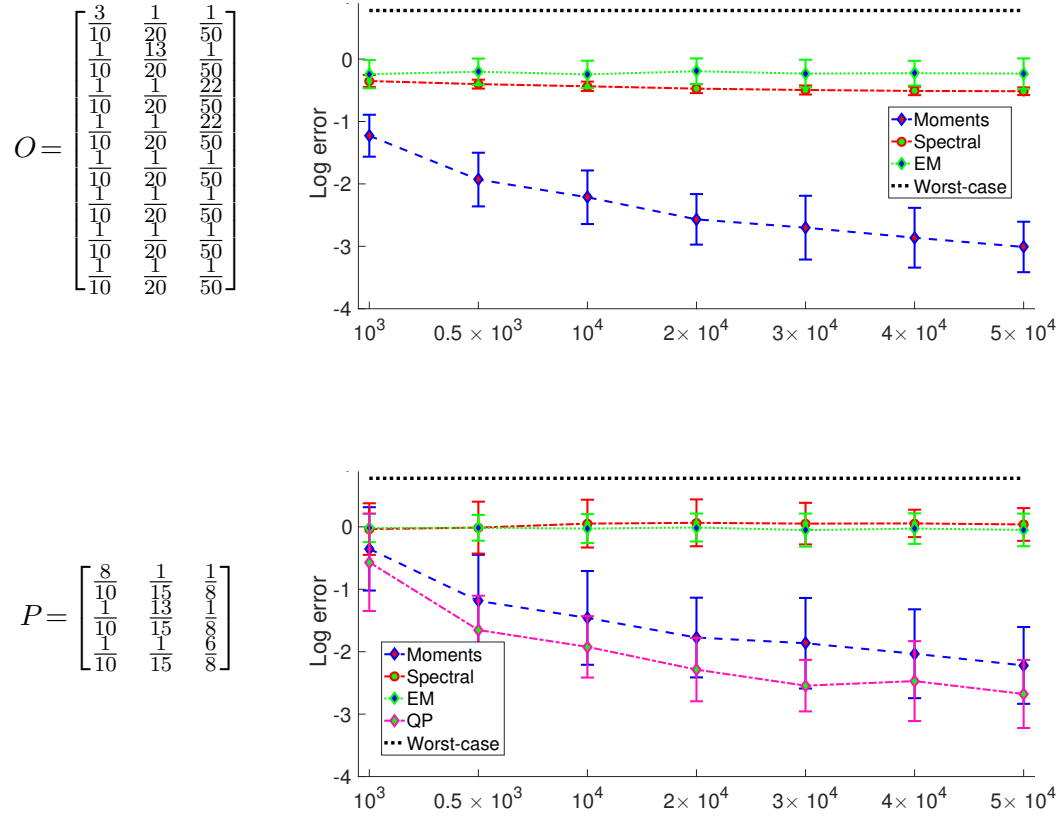


FIGURE 2.3: The mean squared-error in estimating the entire of the emission matrix O (top) and the transition matrix P (bottom) as a function of the number of observations.

entries of the corresponding matrix, averaged over 100 independent runs (the standard deviation of the error is also indicated). This example, taken from Mattfeld (2014), as well as several other experiments, including both discrete- and continuous-output HMMs, which are omitted from this report, demonstrates that Moments has the best performance in estimating the emission matrix, and our novel combined method QP has the best performance in estimating the transition matrix.

2.7.2 Inference on Synthetic Data

In this section we evaluate the performance of our inference algorithm on synthetic data, and compare it to the structured variational inference (SVI) method of Ghahramani & Jordan (1997), the method of Kolter & Jaakkola (2012) and that of Zhong

et al. (2014); we shall refer to the last two algorithms as KJ and ZGS, respectively.

The synthetic dataset used in the experiments was generated in the following way: The power levels corresponding to each on state of each appliance ($\mu_{i,s}, i \in [M], s \in \{2, \dots, K_i\}$) were generated uniformly at random from $[100, 4500]$ with the additional constraint that the difference of any two non-zero levels is greater than 100 (to encourage identifiability), the levels for off states ($\mu_{i,1}$) were set to 0. The transition matrices for each appliance were generated the following way: first the diagonal elements for off states were drawn uniformly at random from $[0, 35]$ and for on states from $[0, 30]$, while non-diagonal elements were selected from $[0, 1]$ to ensure sparse transitions; then the matrices were normalized to ensure they are proper transition matrices. Finally, the output of each appliance was subject to an additive Gaussian noise with variance $\sigma \in [0, 6]$ selected proportionally to the energy consumption level of the given on state, and 1 for off states.¹⁰

The performance of the methods is evaluated through the *normalized disaggregation error*, as suggested by Kolter & Jaakkola (2012); Zhong et al. (2014), which measures the cumulated reconstruction error for the power output of each individual HMM: given the true output $y_{t,i}$ and the estimated output $\hat{y}_{t,i}$ (i.e., $\hat{y}_{t,i} = \mu_i^\top \hat{x}_{t,i}$, where $\hat{x}_{t,i}$ is the estimate of the state-indicator vector $x_{t,i}$ of appliance i), the normalized disaggregation error is defined as

$$NDE = \sqrt{\sum_{t,i} (y_{t,i} - \hat{y}_{t,i})^2 / \sum_{t,i} (y_{t,i})^2}.$$

Figures 2.4 and 2.5 show the performance of the algorithms for changing M , the number of HMMs, and changing K , the number of states. Each plot is a report for $T = 1000$ steps averaged over 100 random realizations, showing the mean and standard deviation of the NDE. Our method ADMM-RR runs ADMM for 2500 iterations, performs the randomized rounding after every 250 iterations, and chooses the result with the highest likelihood. We also report the performance of ADMM followed by simple rounding (i.e., selecting the largest coordinate of $\hat{x}_{t,i}$ for the state $s_{t,i}$). It can be observed that ADMM-RR consistently obtained better results than its competitors,

¹⁰Note that this more realistic model slightly differs from our formal model introduced in Section 2.2, where the noise was added to the total power consumption, and hence the variance of the noise was independent of the number of appliances being used.

ADMM coming second most of the time, followed by KJ and ZGS. Comparing ADMM-RR and ADMM also shows that the repeated randomized rounding has a significant effect on the performance.

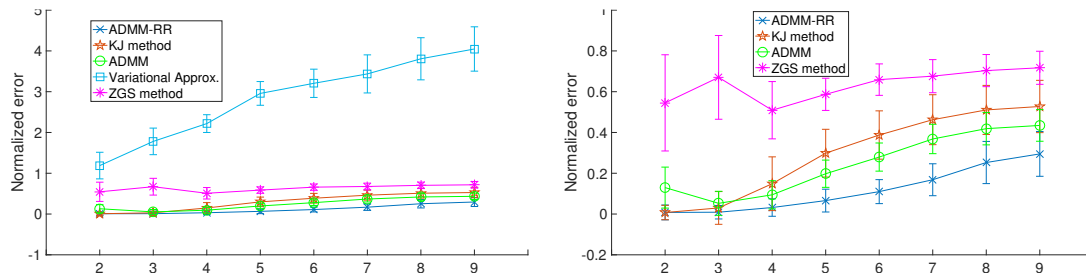


FIGURE 2.4: Disaggregation error varying the number of HMMs. The number of states is fixed and equal to 3. The length of the observation signal is 1000 and the results are averaged over 100 random realizations.

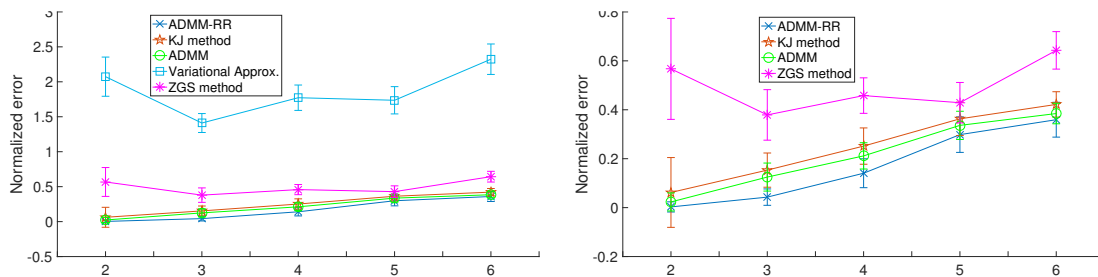


FIGURE 2.5: Disaggregation error varying the number of states. The number of appliances is fixed and equal to 5. The length of the observation signal is 1000 and the results are averaged over 100 random realizations.

2.7.3 Real Data

In this section we compare our method, ADMM-RR, with the best methods from the literature, KJ and ZGS, on two real datasets, REDD (Kolter & Johnson, 2011) and UK-DALE (Kelly & Knottenbelt, 2015b). Note that these datasets contain signals from unmodeled appliances. Since ZGS does not have any mechanism to deal with

this issue and the application of the original form of KJ leads to a poor performance in practice, we augmented both methods with our solution for unmodeled appliances, described in Section 2.6.

For the REDD dataset, we used house No.1 for our experiment. We use the first half of the data for training and the second half for testing. For learning HMMs, the QP algorithm can be applied to both continuous and discrete data. However, the moments method only can work with discrete data. So, we bin the observation data to have a sequence of categorical (discretized) data. The bin bounds are the training data quantiles and the number of quantiles for each training set is $q = 12 \times K$. Each HMM (i.e., appliance) is trained separately using the associated circuit level data, and the HMM corresponding to unregistered appliances is trained using the main panel data.¹¹ In this set of experiments we monitor appliances consuming more than 100 watts (all the other appliances are considered under the “unregistered” category). The ADMM algorithm was run for 1000 iterations, the local search (RR) was run at the end of every 250 iterations, and the result with the largest likelihood was chosen. The experiments were repeated 5 times, averages and the corresponding maximum deviations are reported. To be able to use the ZGS method on this data, we need to have some prior information about the usage of each appliance; the authors suggestion is to use national energy surveys, but in the lack of this information (also about the number of residents, type of houses, etc.) we used the training data to extract this prior knowledge (since this is expected to be more accurate than the survey data, this gives an advantage to ZGS). Since our target function has a number of parameters ($\sigma, \sigma_{\text{diff}}$ in eq. 2.2 and L in Algorithm 2), we tested our method under different parameter settings. We found that the performance of ADMM-RR remained fairly stable over a reasonably large range for the parameter values: varying the parameters in the range $[1, 15]$, for example the error in estimating the energy consumption of the different appliances stays in the range 3.74–4.28%.

Detailed results about the precision and recall of estimating which appliances are on, as well as the error in estimating the total energy usage of different appliances are

¹¹Since the appliances usually have long off-periods between active periods, which dominates the training, to improve the learning performance, we shortened the off-periods during training (so that the total length of off periods is 4 times higher than that of the on periods).

given in Table 2.1; in these experiments, each parameter of ADMM-RR was set to 5, based on our results on the synthetic dataset. In summary, we can see that our method, ADMM-RR, consistently outperformed the others, achieving a weighted-average precision and recall of 84.68% and 78.01% (the weights are selected proportionally to the total energy consumption of each appliance) improving about 23% on the precision of KJ (67.99%) with essentially the same recall (77.91%), and massively outperforming ZGS (32.92/71.01%). Considering the error in assigning the power consumption to different appliances, our method again outperformed its competitors (ADMM-RR: 4.02%, KJ: 5.16%, ZGS: 6.36%).

We also tested our algorithm and the KJ algorithm on the UK-DALE dataset, using 5 days of data for training and 50 days for testing. Table 2.3 shows the detailed precision/recall performance for selected large appliances. Since there are lots of missing data for individual appliances in this dataset (basically, for each day the measurements for some major appliances are missing), it was not possible to determine the consumption of unregistered appliances, hence they are not included in the table. The performance for each appliance is measured on the days where their individual consumptions are available. The results show that in this case ADMM-RR slightly outperforms KJ.

2.8 Conclusions and Future Work

We considered the NILM problem and provided an efficient method based on SDP relaxation, randomized rounding and ADMM for the corresponding FHMM inference problems. By combining two existing methods, we also introduced a new algorithm for estimating the parameters of the individual HMMs based on data. Experiments demonstrate that our method significantly outperforms the state-of-the-art in the NILM problem, and might find other usage in applications based on FHMM models.

With respect to the future development of the NILM algorithm presented in this chapter, we believe that the most beneficial next step would be to improve the modeling of the system's dynamics. In spite of numerous advantages that FHMMs have for modeling home appliances, they are not flexible enough to fully capture the probabilistic nature of appliances' dynamics. To make this clear, consider appliances such

Appliance	Precision/Recall		
	ADMM-RR	KJ method	ZGS method
1 Oven-3	$56.60 \pm 2.36/61.23 \pm 1.54\%$	13.02/64.32%	12.90/38.09%
2 Fridge	$94.70 \pm 0.04/99.30 \pm 0.06\%$	84.98/92.38%	51.39/80.44%
3 Microwave	$80.72 \pm 2.99/39.95 \pm 3.46\%$	14.53/85.34%	8.43/57.59%
4 Bath. GFI-12	$31.87 \pm 2.58/55.39 \pm 3.59\%$	2.37/73.77%	1.36/9.06%
5 Kitch. Out.-15	$59.42 \pm 3.23/94.24 \pm 4.03\%$	5.37/79.58%	8.64/36.69%
6 Wash./Dry.-20-A	$98.26 \pm 0.10/93.60 \pm 1.21\%$	70.41/98.19%	15.49/65.58%
7 Unregistered-A	$93.52 \pm 0.03/98.44 \pm 0.14\%$	92.84/54.44%	57.25/99.33%
8 Oven-4	$10.82 \pm 0.27/28.81 \pm 0.75\%$	6.00/31.72%	2.63/9.16%
9 Dishwasher-6	$56.37 \pm 0.84/94.69 \pm 0.38\%$	26.38/98.11%	29.68/43.26%
10 Wash./Dryer-10	$21.59 \pm 1.10/64.53 \pm 3.37\%$	30.18/40.69%	6.35/4.31%
11 Kitch. Out.-16	$7.74 \pm 0.58/94.66 \pm 3.42\%$	3.82/100%	0/0%
12 Wash./Dry.-20-B	$96.53 \pm 1.11/17.22 \pm 2.17\%$	77.94/71.71%	34.00/67.02%
13 Unregistered-B	$97.73 \pm 0.01/92.56 \pm 0.21\%$	96.07/75.09%	33.63/99.98%
Weighted average	$84.68 \pm 0.28/78.01 \pm 0.35\%$	67.99/77.91%	32.92/71.01%

TABLE 2.1: Disaggregation performance on the REDD dataset: precision/recall and the error in the assigned power to each appliance as a percentage of the total power (for ADMM-RR we report averages over 5 runs and the corresponding maximum deviations). For calculating the weighted averages, the share of each appliance in the total energy consumption is considered as its weight.

Appliance	Error in assigned energy			
	Actual energy	ADMM-RR	KJ	ZGS
1 Oven-3	2.67%	$0.19 \pm 0.15\%$	0.20%	1.47%
2 Fridge	17.45%	$2.06 \pm 0.00\%$	2.63%	0.44%
3 Microwave	4.79%	$2.32 \pm 0.13\%$	0.81%	4.01%
4 Bath. GFI-12	2.10%	$1.00 \pm 0.11\%$	3.84%	0.17%
5 Kitch. Out.-15	1.77%	$0.54 \pm 0.15\%$	4.76%	0.13%
6 Wash./Dry.-20-A	13.54%	$0.76 \pm 0.11\%$	0.79%	7.20%
7 Unregistered-A	14.45%	$0.70 \pm 0.01\%$	6.03%	16.87%
8 Oven-4	3.14%	$7.29 \pm 0.29\%$	6.93%	1.44%
9 Dishwasher-6	8.07%	$3.08 \pm 0.14\%$	7.94%	0.60%
10 Wash./Dryer-10	1.96%	$4.05 \pm 0.02\%$	0.30%	1.48%
11 Kitch. Out.-16	0.27%	$2.71 \pm 0.15\%$	3.35%	0.71%
12 Wash./Dry.-20-B	13.54%	$11.15 \pm 0.23\%$	8.58%	7.20%
13 Unregistered-B	16.17%	$7.62 \pm 0.03\%$	9.16%	9.51%
Total	100%	—	—	—
Weighted average	—	$4.02 \pm 0.06\%$	5.16%	6.36%

TABLE 2.2: Disaggregation performance on the REDD dataset: the error in the assigned power to each appliance as a percentage of the total power (for ADMM-RR we report averages over 5 runs and the corresponding maximum deviations). For calculating the weighted averages, the share of each appliance in the total energy consumption is considered as its weight.

Appliance	ADMM-RR	KJ method
1 Kettle	$98.34 \pm 0.07 / 85.32 \pm 0.90\%$	12.44/82.72%
2 Rice cooker	$21.55 \pm 0.68 / 73.05 \pm 1.26\%$	9.43/63.59%
3 Running Mach.	$3.63 \pm 0.09 / 66.04 \pm 1.03\%$	1.77/99.45%
4 Washing Mach.	$14.06 \pm 0.44 / 9.79 \pm 0.32\%$	18.10/62.66%
5 Dish washer	$61.54 \pm 0.66 / 50.08 \pm 0.51\%$	56.07/63.19%
6 Fridge	$67.31 \pm 0.02 / 92.36 \pm 0.12\%$	93.68/90.47%
7 Microwave	$66.30 \pm 1.66 / 78.49 \pm 1.99\%$	10.07/79.25%
8 Toaster	$9.26 \pm 1.03 / 84.61 \pm 9.89\%$	0.48/96.70%
Weighted average	$63.45 \pm 0.12 / 67.10 \pm 0.30\%$	50.60/76.07%
Energy error	$4.79 \pm 0.11\%$	5.14%

TABLE 2.3: Precision/recall of disaggregation on the UK-DALE dataset.

as a kettle or a toaster which are more likely to be used in the morning, or appliances such as a washing machine or a dryer which are more likely to be used during weekends. This type of information can definitely help the inference algorithm to better identify the appliances. However, FHMM transition matrices, which model the appliances' dynamics, fail to capture such information. Using *inhomogeneous* HMMs instead of vanilla HMMs is one way to address this issue. The idea of using inhomogeneous HMMs has already been explored in (Kim et al., 2011); however, due to the difficulty of inference in such models, Kim et al. (2011) used a simple on/off model for each appliance (i.e., each appliance is modeled with only two states) which adversely affected the algorithm's performance. We believe the inference algorithm proposed in this thesis can be easily extended to be used in such models.

In addition, one of the main assumptions in our model was that appliances evolve independently from each other, which enabled us to use a FHMM for modeling the system. However, in practice, this assumption is not quite true; there are several groups of appliances that are more likely to be used together, such as kitchen appliances. Since NILM is an underdetermined problem, restricting the search space for the inference algorithm is crucial to find a reasonable answer, and this type of information can be helpful in this manner. Therefore, developing learning procedures that can capture such a relationship among appliances is of particular interest in NILM problems. In graphical models terms, these learning procedures should be able to infer the structure of the graph as well as its parameters from the observed data.

Appendix 2.A Discussion of the Derivation of Kolter & Jaakkola (2012) in the Presence of the “Generic Model”

The “generic model” affects the derivation of the algorithm of Kolter & Jaakkola (2012) as follows. The authors claim to derive the final optimization problem given in equation (15) of their paper from (9) and (10) as follows: equation (9) defines the problem $\min_{z \in Z, Q \in A} f_1(z, Q)$, while (10) defines the problem $\min_{z' \in Z', Q \in A} f_2(z', Q)$ where $z' = g(z)$. Here, z, z' are variables that describe the state of the “generic model” over time.

The claim in the paper of Kolter & Jaakkola (2012) is that with some set B (coming from their “one-at-a-time” constraint),

$$\min_{z \in Z, z' \in Z', Q \in A \cap B, z' = g(z)} f_1(z, A) + f_2(z', Q) \quad (2.19)$$

is equivalent to the minimization problem in equation (15). However, carefully checking their derivation shows that (15) is equivalent to $\min_{z \in Z, z' \in Z', Q \in A \cap B} f_1(z, A) + \min_{z' \in Z'} f_2(z', Q)$, which is smaller in general than (2.19).

Chapter 3

Adaptive MCMC via Combining Local Samplers

3.1 Introduction

In this chapter, we consider the problem of computing expectations $\mathbb{E}_P[f(X)] = \int_{\mathcal{X}} f(x)p(x)dx$ for some complicated target distribution P with density p over a set $\mathcal{X} \subset \mathbb{R}^d$ and functions $f : \mathcal{X} \rightarrow \mathbb{R}$. Such expectations are usually intractable, and arise in Bayesian inference and maximum likelihood estimation (Andrieu et al., 2003; Brooks et al., 2011; Gorham & Mackey, 2015). In most practical problems, P has a closed form except for an unknown constant¹ and usually numerical integration is used to approximate the value of the expectation (Neufeld, 2016). Numerical methods fall into two main categories: *deterministic* and *Monte Carlo* (MC) methods. Both approaches approximate the problem by evaluating the function/integrand at some points in the domain $\mathcal{X}$: the former method deterministically chooses these points, while the latter method randomly samples them.

The deterministic methods work well for low-dimensional applications. They have fast convergence rates due to using interpolation between evaluated points which helps with better approximating the integrand. While the interpolation works well for low-dimensional and smooth functions, using it in high-dimensional applications is challenging, as the sample complexity of these methods is typically exponential in the dimension (Neufeld, 2016; Sokal, 1989). For example, for a smooth integrand, the

¹In most real world applications finding this normalization constant is as hard as computing the original expectation under P . For example, when P is obtained as a posterior distribution, p can usually be written as $p(x) = \hat{p}(x) / \int_{\mathcal{X}} \hat{p}(x')dx'$ from Bayes rule, where $\hat{p}(x)$ is available in closed form for any $x \in \mathcal{X}$, but the value of the integral is unknown and is hard to compute.

error in the Simpson rule algorithm converges to zeros as $\frac{1}{n^{4/d}}$ where d is the problem's dimension and n is the number of the nodal points (Sokal, 1989). In contrast, almost all Monte Carlo methods enjoy a convergence rate that is independent of the problem dimension. In its simplest form, MC approximates intractable integrals by $\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n f(x_i)$, where $\{x_i\}_{i=1}^n$ is a sequence of independent and identically distributed (i.i.d.) samples from P .²

Under mild conditions, the error of MC estimators converge to zero at a rate of $\frac{1}{\sqrt{n}}$, which is the direct consequence of the central limit theorem (Sokal, 1989). Given i.i.d. samples $\{x_i\}_{i=1}^n$ from P , $\hat{\mu}_n$ almost surely converges to $\mathbb{E}_P[f(X)]$ as $n \rightarrow \infty$ by the law of large numbers (Andrieu et al., 2003), and if the variance of $f(x)$ is bounded that is $\sigma_f^2 = \mathbb{E}_P[f^2(X) - \mathbb{E}_P[f(X)]^2] < \infty$, then the variance of the estimator (which equals the variance of the error) is $\frac{\sigma_f^2}{n}$ and by using the central limit theorem, one can obtain that

$$\sqrt{n}(\hat{\mu}_n - \mathbb{E}_P[f(X)]) \xrightarrow{d} \mathcal{N}(0, \sigma_f^2) \quad \text{as } n \rightarrow \infty$$

where $\mathcal{N}(\mu, \sigma^2)$ indicates a normal distribution with mean μ and variance σ^2 and $\xrightarrow{d}$ denotes convergence in distribution (Andrieu et al., 2003). This dimensional independent convergence rate gives MC methods a great advantage over their deterministic numerical counterparts in high-dimensional application (Doucet et al., 2001a). However, MC methods usually rely on sampling from an easy-to-sample distribution instead of directly sampling from the target distribution to generate samples, and designing such a proposal distribution for MC methods in high-dimensional problems is challenging by itself (Liu & Lee, 2016). Therefore, Markov chain Monte Carlo (MCMC) methods are used in these applications (Gorham & Mackey, 2015, 2017).

MCMC methods are usually employed for solving inference and optimization problems in “high-dimensional” applications. Similarly to MC methods, MCMC methods enjoy a convergence rate that is independent of the dimensionality of the problem (Doucet et al., 2001a). MCMC methods take random samples from an ergodic Markov

²In the literature, there is an ambiguity in using the terms “Monte Carlo” and “Markov chain Monte Carlo”. While some authors consider Markov chain Monte Carlo methods as a subclass of Monte Carlo methods, others consider them as separate classes of methods. In this thesis, we consider Monte Carlo and Markov chain Monte Carlo as two separate classes of methods: Monte Carlo methods generate i.i.d. samples while Markov chain Monte Carlo methods generate correlated samples.

chain with stationary distribution P , and approximate the expectation $\mathbb{E}_P[f(X)]$ via averaging over the produced sample $\mathbb{E}_{Q_n}[f(Z)] = \frac{1}{n} \sum_{i=1}^n f(x_i)$ where Q_n is the empirical distribution of samples.³ Here, ergodicity means that the chain converges to the same stationary distribution regardless of its starting point (see Section 3.2 for more details).

The challenging problem in designing MCMC methods is to ensure that the distribution of the samples Q_n converge to P fast. Finding a fast-converging Markov chain is crucial since using samples whose empirical distribution is far from the target P can lead to biased results. To design a fast-converging Markov chain, usually some domain specific knowledge is used in the design of their Markov chains. To make this clear, consider Figure 3.1 which presents the results of simulating a Metropolis-Hastings (MH) algorithm with the same stationary distribution and for the same number of samples but different parameters. MH algorithms are the most popular MCMC method (Andrieu et al., 2003) and will be discussed in Section 3.1. The left figure shows samples obtained by a MH algorithm with a proposal function which typically generates small moves:⁴ it can be seen that the chain fails to explore all modes within the given computational budget. In contrast, the right figure corresponds to a MH algorithm with a proposal function which typically generates large moves: in this case the chain explores all the modes, however, its samples are highly correlated. The correlation in this figure can be understood by comparing the number of “dots/samples” in this figure with those in the right figure; the number of dots in the right figure is much lower than the number of dots in the other figure. This means the chain stays in each state for a long time. Obviously, an optimal proposal is somewhere between these two extreme cases.

This need for specialized design led to the development of dozens of methods for each problem, each of which has their own tunable parameters (Neufeld et al., 2014). Consequently, choosing the right method with corresponding parameters to achieve fast convergence is quite difficult and requires considerable time and effort. To address this difficulty, a large body of work has been done: multiple papers proposed ways

³The discrete random variable Z is used to point out that we have summation rather than integral in the calculation of $\mathbb{E}_{Q_n}[f(Z)]$.

⁴The MH algorithm uses an easy-to-sample density function $q(\cdot|x)$ to generate a candidate for the next sample y given the current state of the algorithm, x , and accepts this hypothesis with some probability $\rho(x, y)$. In the literature, the distribution q is called the proposal function.

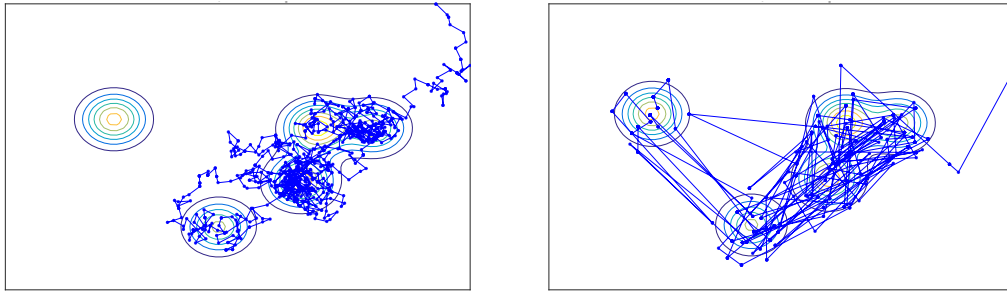


FIGURE 3.1: Simulating a MH algorithm with different proposal functions. The ellipsoids represent the contours of the target distribution. The same number of samples are taken from both proposals. The left figure corresponds to a MH algorithm with small moves, while the right figure corresponds to a MH algorithm with large moves.

for optimal parameter-setting, for instance optimal tuning of the Metropolis-Hasting algorithms was discussed in (Atchadé et al., 2011; Bédard, 2008; Roberts & Rosenthal, 2001; Roberts et al., 1997) (Brooks et al., 2011, Chapter 4). The problem with this line of research is that they rely on some Markov chain parameters (stationary distribution) that are typically unknown (Łatuszyński et al., 2013).

A more promising line of research to address the parameter setting issue is based on adaptive MCMC methods. In this framework, the MCMC samples are used to learn about the target distribution, and the algorithms adjust their parameters as they progress (Łatuszyński et al., 2013). To do so, they rely on optimizing some objective function such as expected squared jumping distance (Pasarica & Gelman, 2010; Wang et al., 2013), the area under the autocorrelation function up to some specific lag (Mahendran et al., 2012), the difference between the proposal covariance matrix and its empirical estimation (Haario et al., 2001, 2006; Mbalawata et al., 2015; Sejdinovic et al., 2014), or the difference between the optimal acceptance rate (in practice, a recommendation thereof) and its empirical estimate (Yang & Rosenthal, 2017). Perhaps the most successful adaptive method for finding the optimal parameters (i.e., the number of leapfrog steps and the step size) in Hamiltonian Monte Carlo (HMC), called no-U-turn sampler (NUTS), is based on monitoring when the sample trajectories “turn back”, and the resulting algorithm provides state-of-the-art performance in a number of problems (Hoffman & Gelman, 2014). For more details about adaptive MCMC

methods, the reader is referred to the tutorial paper by Andrieu & Thoms (2008).

In practice, making sure that a sampler can move between distant modes of the target distribution cannot be guaranteed by the aforementioned adaptive methods.⁵ There are two main approaches to deal with distant modes: (i) running parallel chains and combining their final samples; and (ii) sampling from powers of the target function (the inverse power is referred to as temperature), known as annealing. Indeed, in the literature, there are several successful methods based on combining these two ideas, such as parallel tempering (Earl & Deem, 2005) and (annealed) sequential Monte Carlo samplers, also known as particle filters (Doucet et al., 2001b; Moral et al., 2006). The core idea of these methods and their variants is to run several chains with different temperatures and periodically exchange information between them. By increasing the temperature of a target distribution, it flattens (becomes more uniform), and the chains at high temperatures act as global search algorithms. They pass the information regarding the location of the modes to the chains with lower temperatures, which can effectively search each mode. An alternative to parallelization is *regeneration* (Ahn et al., 2013; Mykland et al., 1995). Regeneration partitions a long Markov chain into smaller independent segments such that the samples are unbiased in each segment, hence can be combined together without any further consideration. This process makes it possible to combine samples from different samplers and tune the parameters of the Markov chain after each regeneration. Although theoretically elegant, the application of regeneration methods is limited in practice since they require a properly tuned distribution for detecting regenerations.

In this chapter we combine the strengths of adaptive and parallel MCMC methods. Instead of trying to find a single very good sampler that approximates the target distribution well on its whole domain, we run several samplers and select which sampler to use at any given time in a *sequential* manner, based on all the samples obtained before. Our main contributions are (i) adapting the bandit-based adaptive Monte Carlo method of Neufeld et al. (2014) for MCMC; and (ii) a novel method for combining samples from multiple chains. The resulting algorithm is suitable for sampling from

⁵To be more precise, instead of modes the “high-probability” regions of the target distribution are of interest, but, for simplicity and following the standard language of MCMC literature, we will often refer to these as modes throughout the paper.

challenging multimodal distributions and is fairly insensitive to the choice of its parameters. The next subsection gives a more detailed overview of our approach and the corresponding challenges.

3.1.1 Approach and Challenges

In the simple case when all MCMC samplers mix well on the whole domain, our goal is to use the best sampler (which mixes the fastest) most of the time. This is very similar to the case of choosing from several unbiased MC samplers. For the latter, Neufeld et al. (2014) showed that scheduling which samplers to use at any point in time is equivalent to a stochastic multi-armed bandit problem (Bubeck & Cesa-Bianchi, 2012). This makes it possible to apply bandit algorithms to select which sampler to use to get the next sample, and the decision depends on the overall performance of the samplers so far, measured by the variance for each sampler. Extending the same idea to the MCMC case is not trivial, since measuring the quality of MCMC samplers is a much harder task. In fact, until recently, there has not been any empirical measure that can monitor the sample convergence. Common MCMC diagnostics such as effective sample size, trace and mean plots, or asymptotic variance assume the chain asymptotically converges to the target distribution, so they cannot detect asymptotic bias (Gorham & Mackey, 2015). To address this issue, Gorham & Mackey (2015) developed an empirical sample-quality measure that can detect non-convergence (or bias) based on Stein’s method. A kernelized version of this measure, called *kernel Stein discrepancy* (KSD) was subsequently developed by Chwialkowski et al. (2016); Gorham & Mackey (2017); Liu et al. (2016), which can be used to compare the quality of different samplers. *As our first contribution, we extend the bandit-based racing method of Neufeld et al. (2014) to MCMC samplers by using the KSD measure as the loss function in the bandit algorithms.* This is described in detail in Section 3.5.1 while the background on KSD is given in Section 3.3.2.

On the negative side, KSD is not able to detect underfitting if the target distribution has well-separated modes (with a distance that is large relative to the kernel width), so it cannot distinguish, between two samplers such that one samples only one

mode while the other samples both modes, and the samples are equally good locally (Chwialkowski et al., 2016; Gorham & Mackey, 2017; Liu et al., 2016). This brings us to the next problem: namely, MCMC methods using a single chain usually fail to explore the whole domain if the support has reasonably high-probability regions separated by low probability regions (of course, such notion of separation depends on the actual sampler used). Setting the parameters of the samplers to deal with this issue and, more importantly, detecting its presence is hard (e.g., it is hard to tune regeneration methods properly), and to our knowledge no practical solutions are available. To alleviate this problem, following the parallel MCMC framework, we run several chains in parallel only expecting that they provide good samples locally. The hope is that the multiple instances will explore the space sufficiently, finding all the important regions of the support. Then, in the end, we combine all the samples (from all the samplers) to approximate the target distribution. This is challenging for two reasons: (i) it is not obvious how the samples from different samplers should be weighted, and (ii) we do not want to waste resources to run several samplers exploring the same region of the domain. For (ii), we apply our bandit-based racing method locally (Section 3.5.2), while to address (i), *we develop a method to estimate the probability of the region a set of samples cover based on Rényi-entropy estimates (Pál et al., 2010), and use these to weight the samples, which is our second main contribution.* This is described in Section 3.5.3.

Our final sampling algorithm is put together in Section 3.6. Lastly, in Section 3.7 we demonstrate through a number of experiments that our method is competitive with state-of-the-art adaptive MCMC methods, such as the no-U-turn sampler NUTS (Hoffman & Gelman, 2014) or the recent sample reweighting method of Liu & Lee (2016) on simpler cases when the distribution is concentrated on a single “connected” region, while significantly outperforming the competitors, including parallel tampering and sequential MC, on harder problems where high-probability regions are separated by areas of low-probability, as well as on a challenging sensor-localization problem.

Before describing our contributions in detail, we first provide the necessary background by introducing MCMC methods, the sample quality measure of interest, and

the multi-armed bandit problem in Section 3.1, Section 3.3 and Section 3.4, respectively.

3.2 Background: Markov Chain Monte Carlo Methods

Similarly to Monte Carlo integration methods, MCMC methods try to approximate $\mathbb{E}_P[f(X)] = \int_{\mathcal{X}} f(x)p(x)dx$ by the empirical average $\mathbb{E}_{Q_n}[f(Z)] = \frac{1}{n} \sum_{t=1}^n f(x_t)$ for samples $(X_1, \dots, X_n)$ which are generated sequentially according to a Markov process with stationary distribution P . Formally, an MCMC method for sampling from a target density p is a procedure that generates samples from an ergodic Markov chain (X_t) with stationary distribution P . Evolution of the chain (X_t) is governed by the *transition kernels* K ; $K(X_t, X_{t+1})$ determines the conditional density of the $(t+1)$ -th sample X_{t+1} given the t -th sample X_t (i.e., $X_{t+1} \sim K(X_t, \cdot)$). A typical example of a Markov chain is a random walk process. Formally, a sequence of random variables (X_t) is a random walk process if it satisfies

$$X_{t+1} = X_t + \epsilon_t,$$

where for all t , the random variable ϵ_t is generated independently of $X_t, X_{t-1}, \dots, X_1$. The difference between different MCMC methods is in the way they construct the transition kernel.

A Markov chain needs to be ergodic in order to be qualified as an MCMC sampler. Ergodicity ensures that the empirical distribution of the samples converges to a unique stationary distribution, independently of its initial state X_0 . A Markov chain is ergodic if its *irreducible* and *aperiodic*. Irreducibility ensures each state in the problem domain can be reached in a finite number of steps from any other state. Aperiodicity ensures that the chain does not get trapped in cycles (Andrieu et al., 2003; Robert & Casella, 2005). In what follows, we formally define some of these properties of Markov chains.⁶

⁶The material for rest of the section is mainly from Robert & Casella (2005) unless otherwise stated.

Irreducibility

The irreducibility of a Markov chain ensures that each state in the problem domain can be reached in a finite number of steps from any other state. Let $\mathcal{B}(\mathcal{X})$ be a σ -algebra on a set $\mathcal{X}$ and $K^n(x, A) = \int_{\mathcal{X}} K^{n-1}(y, A)K(x, dy)$ for $n > 1$ be the kernel for n transitions. Then irreducibility is defined as follows:

Definition 3.1. (Robert & Casella, 2005, Definition 6.13) Given a measure φ on $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$, the Markov chain (X_t) with transition kernel K is φ -irreducible if, for every $A \in \mathcal{B}(\mathcal{X})$ with $\varphi(A) > 0$, there exists t such that $K^t(x, A) > 0$ for all $x \in \mathcal{X}$. The chain is strongly φ -irreducible if $t = 1$ for all measurable A .

Aperiodicity

As mentioned earlier, aperiodicity ensures that the chain (X_t) is not periodic and it does not get trapped in any cycle. An irreducible chain is aperiodic if it has period $d = 1$ where the period of (X_t) is defined as the largest integer d satisfying Definition 3.3.

Definition 3.2. (Robert & Casella, 2005, Definition 6.19) A set C is small if there exist $m \in \mathbb{N}^+$ and a nonzero measure ν_m over $\mathcal{X}$ such that

$$K^m(x, A) \geq \nu_m(A), \quad \forall x \in C, \forall A \in \mathcal{B}(\mathcal{X}).$$

Definition 3.3. (Robert & Casella, 2005, Definition 6.23) A ψ -irreducible chain (X_t) has a *cycle of length* d if there exists a small set C , an associated integer M , and a probability distribution ν_M such that d is the greatest common divisor of

$$\{m \geq 1; \exists \delta_m > 0 \text{ such that } C \text{ is small for } \nu_m \geq \delta_m \nu_M\}.$$

Harris Recurrence

So far, we have discussed that irreducibility guarantees that every set A in the state space is visited by a Markov chain; however, this property does not ensure that the

chain is in A infinitely many times. The notion of *recurrence* formalizes this form of Markov chains' stability.

Let $\eta_A = \sum_{t=1}^{\infty} \mathbb{I}_A(X_t)$ denote the total number of times X_t visits the set A . Then recurrence is defined the following way:

Definition 3.4. (Robert & Casella, 2005, Definition 6.29) A Markov chain (X_t) is recurrent if

- (i) there exists a measure ψ such that (X_t) is ψ -irreducible, and
- (ii) for every $A \in \mathcal{B}(\mathcal{X})$ such that $\psi(A) > 0$, $\mathbb{E}[\eta_A | X_1 = x] = \infty$ for every $x \in A$.

A ψ -irreducible chain is either recurrent or transient; in the later case $\mathbb{E}[\eta_A | X_1 = x]$ is finite for some A with $\psi(A) > 0$. A stronger notion of chain stability can be established by requiring $P(\eta_A = \infty | X_1 = x) = 1$ instead of $\mathbb{E}[\eta_A | X_1 = x] = \infty$ which is called *Harris recurrence*.

Definition 3.5. (Robert & Casella, 2005, Definition 6.32) A set A is *Harris recurrent* if $P(\eta_A = \infty | X_1 = x) = 1$ for all $x \in A$. The chain (X_t) is Harris recurrent if there exists a measure ψ such that (X_t) is ψ -irreducible and for every A with $\psi(A) > 0$, A is Harris recurrent.

Stationary Chain

A strong notion of stability can be defined for a Markov chain (X_t) by demanding the marginal distribution of X_t to be independent of t which means $X_t \sim \pi$ implies $X_{t+1} \sim \pi$. Indeed, the core idea behind MCMC methods is that we can have a Markov chain that meets this requirement. This time-independent marginal distribution is called *invariant* distribution (a.k.a. *stationary* distribution). In other words,

Definition 3.6. (Robert & Casella, 2005, Definition 6.35) A σ -finite measure π is invariant for the transition kernel $K(\cdot, \cdot)$ (and for the associated chain) if

$$\pi(B) = \int_{\mathcal{X}} K(x, B) \pi(dx), \quad \forall B \in \mathcal{B}(\mathcal{X}).$$

A ψ -irreducible chain is *positive*, if it has an invariant probability measure. On the other hand, recurrent chains that do not have an invariant probability measure are called *null recurrent*.

Reversibility and the Detailed Balance Condition

Reversibility requires the dynamics of the chain to be independent of the direction of time.

Definition 3.7. (Robert & Casella, 2005, Definition 6.44) A stationary Markov chain (X_t) is reversible if the conditional distribution of X_{t+1} given $X_{t+2} = x$ is the same as the conditional distribution of X_{t+1} given $X_t = x$.

A Markov chain with transition kernel $K(\cdot, \cdot)$ satisfies the *detailed balance condition* if there is a density π such that the following equality holds for every pair $x, y \in \mathcal{X}$:

$$K(y, x)\pi(y) = K(x, y)\pi(x).$$

The detailed balance requires that the likelihood of being in x and moving to y is equal to the likelihood of being in y and moving to x . The following theorem shows the connection between the detailed balance condition, reversibility, and the invariant distribution:

Theorem 3.8. (Robert & Casella, 2005, Theorem 6.46) *Suppose that a Markov chain with transition kernel K satisfies the detailed balance condition with a probability density function π . Then the density π is the invariant density of the chain, and the chain is reversible.*

Note that satisfying the detailed balance is not a necessary but a sufficient condition for π to be the invariant density of K . However, since it is easy to check it in practice, it is used for most of MCMC methods.

Ergodicity and Convergence

So far we have discussed different notions of stability of Markov chains and some related properties. Next we consider the limiting behavior of X_t as $t \rightarrow \infty$. Basically, we want to establish sufficient conditions ensuring that the distribution of X_t , or the empirical distribution of the sample $\{X_1, \dots, X_t\}$ converges to the invariant distribution. Indeed, the following theorems explain why we can use MCMC methods in practice for sampling from probability distributions.

For the purpose of sampling, *ergodicity* is the condition that we are mainly interested in. The following theorem shows that being positive Harris recurrent and aperiodic is sufficient for a chain to be ergodic. In the following theorem, $\|\mu_1 - \mu_2\|_{TV} = \sup_A |\mu_1(A) - \mu_2(A)|$ is the total variation distance of two distributions μ_1 and μ_2 .

Theorem 3.9. (Robert & Casella, 2005, Theorem 6.51) *If (X_t) is positive Harris recurrent and aperiodic, then*

$$\lim_{t \rightarrow \infty} \left\| \int K^t(x, \cdot) \mu(dx) - \pi \right\|_{TV} = 0$$

for every initial distribution μ .

For a Markov chain (X_t) and for any function $h : \mathcal{X} \rightarrow \mathbb{R}$, define the partial sum $S_t(h)$ as

$$S_t(h) \triangleq \frac{1}{t} \sum_{i=1}^t h(X_i)$$

for all t . Also, let $L^1(\pi)$ to denote a set of measurable functions f on $\mathcal{X}$ such that $\int_{\mathcal{X}} |f(x)| \pi(dx) < \infty$. We have the following version of the law of large numbers for Markov chains:

Theorem 3.10. (Robert & Casella, 2005, Theorem 6.63) *If (X_t) has a σ -finite invariant measure π , the following two statements are equivalent:*

(i) *If $f, g \in L^1(\pi)$ with $\int g(x) d\pi(x) \neq 0$, then*

$$\lim_{t \rightarrow \infty} \frac{S_t(f)}{S_t(g)} = \frac{\int f(x) d\pi(x)}{\int g(x) d\pi(x)} \quad a.s.$$

Algorithm 4 Metropolis-Hastings algorithm (Robert & Casella, 2005)

Given: Starting point x_1 .
for $t = 1, \dots, n$ **do**
 Generate $Y_t \sim q(\cdot|x_t)$.
 Calculate $\rho(x, Y_t) = \min \left\{ 1, \frac{p(Y_t) q(x|Y_t)}{p(x) q(Y_t|x)} \right\}$.
 Set $x_{t+1} = \begin{cases} Y_t & \text{with probability } \rho(x_t, Y_t), \\ x_t & \text{with probability } 1 - \rho(x_t, Y_t), \end{cases}$
end for

(ii) The Markov chain (X_t) is Harris recurrent.

3.2.1 The Metropolis-Hastings Algorithm

A prominent MCMC method is the *Metropolis-Hastings* (MH) algorithm which is shown in Algorithm 4 (Robert & Casella, 2005). The MH algorithm uses an easy-to-sample density function $q(\cdot|x)$ to generate a candidate for the next sample y given the current state of the algorithm, x , and accepts this hypothesis with probability $\rho(x, y)$ (defined in Algorithm 5). In the literature, the distribution q is called the proposal distribution⁷ and the probability $\rho(x, y)$ is known as the *acceptance probability* (Robert & Casella, 2005). The transition kernel of this algorithm is

$$K(x, y) = \rho(x, y)q(y|x) + (1 - r(x))\delta_x,$$

where $r(x) = \int \rho(x, y)q(y|x)dy$ and δ_x is a Dirac mass at x . Under some nonrestrictive conditions on p and q , the invariant density of the Markov chain generated by Algorithm 4 is p (Andrieu et al., 2003; Robert & Casella, 2005). Unsurprisingly, these conditions are aperiodicity and irreducibility. By considering the MH rejection process, the aperiodicity of the MH Markov chain is guaranteed by the accept-reject step; also, to guarantee irreducibility, it is sufficient to ensure that the support of p is included in the support of q (Andrieu et al., 2003).

More precisely, the following convergence guarantees can be given for the Metropolis-Hastings algorithm:

⁷In practice, often a multivariate normal distribution centered at x is used as $q(\cdot|x)$.

Theorem 3.11. (Robert & Casella, 2005, Theorem 7.4) *Suppose that the Metropolis-Hastings Markov chain (X_t) is p -irreducible.*

(i) *If $h \in L^1(p)$, then*

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T h(X_t) = \int h(x)p(x)dx \quad p\text{-almost surely.}$$

(ii) *If, in addition, (X_t) is aperiodic as well, then*

$$\lim_{t \rightarrow \infty} \left\| \int K^t(x, \cdot) \mu(dx) - p \right\|_{TV} = 0$$

for every initial distribution μ of X_0 , where $K^t(x, \cdot)$ denotes the kernel for t transitions.

3.2.2 Langevin Algorithms

The Metropolis-Hastings algorithm discussed in the previous section relies on a random walk process for exploring the state space and is often referred to as random-walk MH. An alternative exploration paradigm can be developed from *diffusion theory* (Robert & Casella, 2005). The main issue with random walk processes is that they explore the state space slowly, and consequently lead to highly correlated samples. To address this issue of slow exploration, Langevin algorithms replace the random walk search with a faster gradient-guided exploration (Neal, 2012).

To this end, Langevin algorithms produce a *diffusion* process with stationary density p . Instead of a diffusion, any other *stochastic differential equation* producing a continuous-time process with stationary density p can be used. The so-called *Langevin diffusion* with stationary density p is defined by the differential equation

$$dX_t = dB_t + \frac{1}{2} \nabla \log p(X_t) dt, \quad (3.1)$$

where B_t is a standard *Brownian motion*. A Brownian motion B_t is a continuous-time stochastic process such that $B_0 = 0$, $B_t \sim \mathcal{N}(0, w^2 t I_d)$, $B_t - B_{t'} \sim \mathcal{N}(0, w^2 |t - t'| I_d)$,

$B_t - B_{t'}$ is independent of $B_{t'}$ for all $t > t'$, I_d is the identity matrix of size d (Robert & Casella, 2005). A continuous time process such as (3.1) cannot be simulated in computer, so Langevin algorithms use a discretized version of the process. The following equation shows a discrete-time version of (3.1):

$$x_{t+1} = x_t + \frac{\sigma^2}{2} \nabla \log p(x_t) + \sigma \epsilon_t, \quad (3.2)$$

where $\epsilon_t \sim \mathcal{N}(0, I_d)$, and σ^2 is the integration step size (Robert & Casella, 2005). Note that (3.2) depends on the density p through the derivative of $\log p(x)$, so it can be calculated even for the unnormalized density $\hat{p} = cp$. Ideally, the hope is that (3.2) leads to a similar asymptotic behavior as (3.1). However, according to Robert & Casella (2005), such a discrete approximation can have a very different asymptotic behavior than the original continuous diffusion process (i.e., it might not converge to the same invariant distribution as (3.1)). In particular, Robert & Casella (2005) mention that these approximations could be non-convergent, no matter how small the step size σ is. A way to fix this issue and have a correct sampler based on diffusion is to add a Metropolis accept-reject step which ensures p is the stationary density of the discrete process. The corresponding algorithm is called Metropolis-adjusted Langevin algorithm (MALA). MALA generates a new proposal random variable $Y_{t+1} = x_t + \frac{\sigma^2}{2} \nabla \log f(x_t) + \sigma \epsilon_t$ and accepts it with probability

$$\rho(x_t, Y_{t+1}) = \min \left\{ 1, \frac{p(Y_{t+1})}{p(x)} \frac{q(x_t, Y_{t+1})}{q(Y_{t+1}, x_t)} \right\}$$

where

$$q(x, y) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp \left(-\frac{1}{2\sigma^2} \left\| y - x - \frac{\sigma^2}{2} \nabla \log p(x) \right\|_2^2 \right).$$

According to Robert & Casella (2005), MALA samplers are guaranteed to be ergodic, and hence they are asymptotically unbiased in the sense that the empirical distribution of the samples converges to the target distribution. The drift term, $\sigma^2 \nabla \log p(x_t)$, in (3.2) guides the proposal for MALA samplers based on the gradient information. However, if there is a strong correlation between the variables and big difference in their associated variances, using a diffusion/drift which uses constant coefficient σ in all directions might be inefficient. This can be avoided by considering a *pre-conditioning*

matrix M such that $x_{t+1} = x_t + \frac{\sigma^2}{2} M \nabla \log f(x_t) + \sigma \sqrt{M} \epsilon_t$, where the role of M is to decorrelate and normalize the components of x_t (Girolami & Calderhead, 2011).

3.2.3 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) (a.k.a hybrid Monte Carlo) algorithms are another group of sampling algorithms that tend to avoid the slow exploration of random walks by simulating a *Hamiltonian* dynamics, which is a mathematical formalism that describes the evolution of a physical system over time (Girolami & Calderhead, 2011). Hamiltonian dynamics operates on a position variable $x \in \mathbb{R}^d$ and a *momentum* variable $w \in \mathbb{R}^d$, so the dimension of the joint state space is twice the size of each variable. A Hamiltonian $H(x, w)$ shows the total energy of a system, and is defined as

$$H(x, w) = U(x) + K(w), \quad (3.3)$$

where $U(x)$ is called the *potential energy* and $K(w)$ is called the *kinetic energy*. For HMC, $U(x)$ is defined to be the negative log probability density of the target distribution, and $K(w)$ is usually defined as

$$K(w) = \frac{1}{2} w^\top M^{-1} w$$

where M is a symmetric positive-definite matrix, called the mass matrix, which is often a scalar multiple of the identity matrix. Defining $K(w)$ in this way corresponds to the negative log probability density of the a zero-mean Gaussian distribution with covariance matrix M (Neal, 2012).

The partial derivatives of the Hamiltonian H describe how the kinetic and potential energy evolve over time for an object that moves in a system:

$$\frac{\partial x}{\partial t} = \frac{\partial H}{\partial w}, \quad \frac{\partial w}{\partial t} = -\frac{\partial H}{\partial x}.$$

For a given target density p that we wish to sample from, the Hamiltonian in (3.3) becomes

$$H(x, w) = -\mathcal{L}(x) + \frac{1}{2} w^\top M^{-1} w + C \quad (3.4)$$

where $\mathcal{L}(x) \triangleq \log p(x)$ and C is a constant, while its partial derivatives become

$$\frac{\partial x}{\partial t} = \frac{\partial H}{\partial w} = M^{-1}w, \quad \frac{\partial w}{\partial t} = -\frac{\partial H}{\partial x} = -\nabla \mathcal{L}(x). \quad (3.5)$$

Any solution of (3.5) has to satisfy the following conditions: (i) conservation of the Hamiltonian (i.e., preserving the total energy); (ii) volume preservation; (iii) time reversibility. We do not formally define these condition here, but more details can be found in the book of Neal (2012). For most practical applications, these differential equations cannot be solved analytically and need to be solved numerically (Girolami & Calderhead, 2011). According to Girolami & Calderhead (2011), there is a class of numerical solutions for Hamiltonian systems that fully satisfies (ii) and (iii), and approximately satisfies (i). The Stormer-Verlet or *leapfrog* integrator is one of these solutions and can be described as

$$\begin{aligned} w_{t+\frac{\epsilon}{2}} &= w_t + \frac{\epsilon}{2} \nabla \mathcal{L}(x_t); \\ x_{t+\epsilon} &= x_t + \epsilon M^{-1} w_{t+\frac{\epsilon}{2}}; \\ w_{t+\epsilon} &= w_{t+\frac{\epsilon}{2}} + \frac{\epsilon}{2} \nabla \mathcal{L}(x_{t+\epsilon}). \end{aligned} \quad (3.6)$$

As mentioned before, with the above discretization, the total energy is only approximately conserved. To correct this issue, a Metropolis accept-reject step can be added to the final algorithm. By combining the first and second steps in (3.6) in a single step, we get the update equation

$$x_{t+\epsilon} = x_t + \frac{\epsilon^2}{\epsilon} M^{-1} \nabla \mathcal{L}(x_t) + \epsilon M^{-1} w_t.$$

This update equation is fairly similar to the one in (3.2) and, in fact, it is the pre-conditioned Langevin diffusion mentioned earlier (Girolami & Calderhead, 2011). The leapfrog algorithm simulates the Hamiltonian dynamics for a time interval of length ϵ . However, the simulation time can be extent to $L \times \epsilon$ by running the leapfrog algorithm L times, which is often the case in HMC algorithms. A classical HMC algorithm is given in Algorithm 5.

The performance of the HMC algorithms is fairly sensitive to the parameters of the discretization—the number of leapfrog steps L and the step size ϵ . For example, if L

Algorithm 5 Hamiltonian Monte Carlo (Hoffman & Gelman, 2014)

Given: $x_0, \epsilon, L, \mathcal{L}(x), M, T$.

for $t = 1, 2, \dots, T$ **do**

 Sample $w_0 \sim \mathcal{N}(0, I)$.

 Set $x_t = x_{t-1}, \tilde{x} = x_{t-1}, \tilde{w} = w_0$.

for $i = 1, 2, \dots, L$ **do**

 Set $\tilde{x}, \tilde{w} \leftarrow \text{Leapfrog}(\tilde{x}, \tilde{w}, \epsilon)$.

end for

 With probability $\rho = \min \left\{ 1, \frac{\exp(\mathcal{L}(\tilde{x}) - \frac{1}{2}\tilde{w}^\top M^{-1}\tilde{w})}{\exp(\mathcal{L}(x_{t-1}) - \frac{1}{2}w_0^\top M^{-1}w_0)} \right\}$, set $x_t = \tilde{x}$ and $w_t = -\tilde{w}$.

end for

function $\text{Leapfrog}(x, w, \epsilon)$

$\tilde{w} = w + \frac{\epsilon}{2} \nabla \mathcal{L}(x)$.

$\tilde{x} = x + \epsilon M^{-1} \tilde{w}$.

$\tilde{w} = \tilde{w} + \frac{\epsilon}{2} \nabla \mathcal{L}(\tilde{x})$.

return $\tilde{x}, \tilde{w}$

is too small, the exploration process becomes a random-walk, and if it is too large, the algorithm wastes computational resources (Hoffman & Gelman, 2014). There are several algorithms successfully tuning ϵ in the adaptive MCMC literature, but setting L is more challenging and requires expert knowledge (Hoffman & Gelman, 2014). The no-U-turn sampler (NUTS) of Hoffman & Gelman (2014) is an extension of the original HMC algorithm and automatically sets L . For setting L , NUTS works in phases and uses a recursive algorithm to simulate the Hamiltonian dynamics for a set of candidate points, $L = 2^i$ for $i \in \mathbb{N}$. At phase i , it simulates the dynamics for $L = 2^i$ steps in both directions (i.e., forward and backward in time) and continues this process until the derivative of the squared distance between the current candidate point $\tilde{x}$ and initial position x ($\frac{d}{dt}(\tilde{x} - x)^\top(\tilde{x} - x)$) becomes negative. NUTS can also tune the step size ϵ based on dual averaging which enables it to be run without any hand-tuning.

Finally, HMC algorithms—including NUTS—satisfy the detailed balance condition, and hence they are asymptotically unbiased in the sense that the empirical distribution of the samples converges to the target distribution (Hoffman & Gelman, 2014; Neal, 2012).

3.2.4 Sequential Monte Carlo Sampler Algorithms

The sequential Monte Carlo sampler is an algorithm to sample *sequentially* from a sequence of probability measures that are defined over the same state space $\mathcal{X}$ (Moral et al., 2006). The algorithm approximates each probability distribution with a group of weighted samples which are propagated through the sequence of distributions by using sequential Monte Carlo methods. In general, running MCMC method in parallel is a difficult task: The SMC sampler algorithm provides a framework for developing simple parallel MCMC methods. The resulting algorithms can be used for global optimization and sequential Bayesian estimation (Moral et al., 2006).

Let $\{p_n\}_{n \in [M]}$ be a set of M probability densities that are defined over the common state space $\mathcal{X}$, and let $\{\hat{p}_n\}_{n \in [M]}$ be a set of M unnormalized probability densities that are defined as $\hat{p}_n = p_n Z_n$ where Z_n is the normalization constant and unknown. Except for p_1 , for which we assume that direct sampling is tractable, we also assume that directly sampling from the rest of the distributions is infeasible. The task is to sequentially sample from $\{p_n\}_{n \in [M]}$ starting from p_1 . Such a problem arises in many global optimization and sequential Bayesian estimation problems (Moral et al., 2006). Such a sequence of distributions $\{p_n\}_{n \in [M]}$ arises for example from applying a simulated annealing procedure to a probability density p with $p_n(x) \propto p(x)^{\Phi_n}$ (where $\propto$ means *propositional to*) for a schedule $\{\Phi_n\}_{n \in [M]}$; or from changing a tractable density μ to the target density p through a sequence of intermediate densities, $p_n(x) \propto p(x)^{\Phi_n} \mu(x)^{1-\Phi_n}$ for $0 \leq \Phi_1 < \dots < \Phi_M = 1$ (Moral et al., 2006).

The overall structure of the sequential MC (SMC) sampler algorithm is fairly similar to *sequential importance sampling*, except the way it calculates the importance weights (Moral et al., 2006). A population of N weighted samples $\{q_1^{(i)}, X_1^{(i)}\}$ is generated from the easy-to-sample distribution p_1 . In the SMC literature, the $X_n^{(i)}$ are called particles. More generally, given population $\{X_n^{(i)}\}$, the next population of particles $\{X_{n+1}^{(i)}\}$ is generated approximately from $\hat{p}_{n+1}$ by applying an MCMC algorithm with invariant distribution $\hat{p}_{n+1}$ for a certain number of steps starting from each point $X_n^{(i)}$ for $i \in [N]$. The importance weights $q_{n+1}^{(i)}$ for $X_{n+1}^{(i)}$ are calculated in a special way that will be explained later. Then, the new population is carried forward through the remaining

distributions by repeating the previous steps. Due to sample degeneracy, the weighted empirical distribution Q_n of the particles might diverge from p_n as n increases. To solve this issue, the particles are subject to a so-called *resampling* procedure, which generates N new particles from the current particles by replicating some of them. A common resampling method is to draw N samples independently from a multinomial distribution choosing $X_n^{(i)}$ with probability $q_n^{(i)}$ (Moral et al., 2006). The sample degeneracy can be measured by calculating the effective sample size (ESS) which is defined as $\text{ESS} = \sum_{i=1}^N 1/[q_n^{(i)}]^2$ and takes values between 1 and N . The pseudocode for an SMC sampler is given in Algorithm 6.

The only remaining ingredient, which in fact is the most important part, is to show how we can calculate the importance weights $q_n^{(i)}$. This can be calculated sequentially from the previous weight as

$$\hat{q}_n = q_{n-1} \tilde{q}_n(x_{n-1}, x_n)$$

where

$$\tilde{q}_n(x_{n-1}, x_n) = \frac{\hat{p}_n L_{n-1}(x_n, x_{n-1})}{\hat{p}_{n-1} K_n(x_{n-1}, x_n)}. \quad (3.7)$$

In this equation, $K_n(x_{n-1}, x_n)$ is the MCMC transition kernel and $L_{n-1}(x_n, x_{n-1})$ is an artificial backward (in time) Markov kernel. Moral et al. (2006) have shown that there is an optimal backward kernel in the sense that it minimizes the variance of the unnormalized importance weights $\hat{q}_n^{(i)}$. However, for most practical problems, this optimal kernel is intractable. So, they recommended a number of suboptimal kernels, see Moral et al. (2006) for more details. Based on these recommendations, in this work we use

$$L_n(x_n, x_{n-1}) = \frac{p_n(x_{n-1}) K_n(x_{n-1}, x_n)}{p_n(x_n)}$$

which is a good approximation to the optimal kernel if $p_{n-1} \approx p_n$ and leads to

$$\tilde{q}_n(x_{n-1}, x_n) = \frac{\hat{p}_n(x_{n-1})}{\hat{p}_{n-1}(x_{n-1})}. \quad (3.8)$$

Note that $\tilde{q}_n$ in (3.8) only uses unnormalized densities, so it can be easily calculated in practice.

Algorithm 6 Sequential Monte Carlo Sampler (Moral et al., 2006)

Given: Sequence of unnormalized densities $\{\hat{p}_n\}_{n \in [M]}$; threshold τ ; number of particles N .
 Sample $X_1^{(i)} \sim \hat{p}_1$ for all $i \in [N]$; Set $q_1^{(i)} = \frac{1}{N}$.
for $n = 2, \dots, K$ **do**
 Calculate ESS for $\{q_n^{(i)}\}_{i \in [N]}$.
 if $\text{ESS} < \tau$ **then**
 Re-sample the last set of particles $\{X_{n-1}^{(i)}\}_{i \in [N]}$.
 Set $q_n^{(i)} = \frac{1}{N}$ for all $i \in [N]$.
 end if
 Sample $X_n^{(i)} \sim K_n(X_{n-1}^{(i)}, \cdot)$ for all $i \in [N]$.
 Calculate $\{\tilde{q}_n^{(i)}\}_{i \in [N]}$ by using (3.8).
 Normalize the weight $q_n^{(i)} = \frac{q_{n-1}^{(i)} \tilde{q}_n^{(i)}}{\sum_{j=1}^N q_{n-1}^{(j)} \tilde{q}_n^{(j)}}$.
end for

3.2.5 Parallel Tempering

Parallel tempering (or replica exchange) methods are complementary to any MCMC algorithms. Their core idea is to run M MCMC methods in parallel and exchange information among them for better exploration of the state space. Each MCMC chain is simulated in a scaled copy (a.k.a. replica) of the target system (i.e., $p_n(x) \propto p(x)^{1/T_n}$ for some temperatures $\{T_n\}_{n \in [M]}$) (Earl & Deem, 2005). The high temperature probability distributions are flatter than the original distribution and the chains associated with them have a better chance of exploring the state space without being trapped in a local mode. Thus, these chains work as global search algorithms and pass the information regarding high probability areas to the chains with lower temperatures: these chains provide precise sampling (i.e., local search) (Earl & Deem, 2005).

For a fixed number of samples per chain, simulation of K chains requires K -times more computational budget, which in turn requires the algorithm to be K -times more efficient compared to running a single chain with the original temperature (i.e., $T = 1$). In practice, it has been observed that parallel tempering usually satisfies this requirement (Earl & Deem, 2005).

Formally, let p be the probability density function that we want to sample from (i.e., $p : \mathcal{X} \rightarrow \mathbb{R}^+$), and let $\{p_n\}_{n \in [M]}$ be a set of replicas in different temperatures,

$p_n(x) \propto e^{\frac{CU(x)}{T_n}}$ where $U(x) = \log p(x)$ is the potential energy, C is a constant and $T_1 < T_2 < \dots < T_M$; normally, $T_1 = 1$. Parallel tempering proceeds as follows: M Markov chains are simulated in parallel, one for each temperature value. After a certain number of steps, adjacent chains (corresponding to T_i and T_{i+1} for $i \in [M-1]$) exchange information by swapping their samples. Such a swap is accepted with probability

$$\rho(x_i^t, x_j^t) = \min \left\{ 1, \exp \left(\left(\frac{C}{T_i} - \frac{C}{T_j} \right) (U(x_i^t) - U(x_j^t)) \right) \right\}.$$

Parallel tempering is an exact method in the sense that it satisfies the detailed balance condition with $p(x)$ as its stationary density, hence is asymptotically consistent (Earl & Deem, 2005).

3.3 Measuring Sample Quality

As mentioned in Section 3.1.1, measuring the quality of samples produced by an MCMC algorithm is crucial in our approach. To this end, we are going to use the recently introduced *kernel Stein discrepancy* (KSD) (Chwialkowski et al., 2016; Gorham & Mackey, 2017; Liu et al., 2016).

To measure the quality of a set of samples, we want to quantify how well a probability distribution Q_n over $\mathcal{X} = \mathbb{R}^d$ can approximate the target distribution P ; in this section we assume that the density p of P is positive and differentiable on the whole $\mathbb{R}^d$. Q_n is given by a weighted sample $\{(x_i, q_i)\}_{i \in [n]}$, where $x_i \in \mathbb{R}^d$ and $Q_n(x_i) = q_i$ with $\sum_{i \in [n]} q_i = 1$. One way to do this is to measure the maximum difference of expectations over a class of real-valued test functions $\mathcal{H} \subset \{h : \mathbb{R}^d \rightarrow \mathbb{R}\}$:

$$\mathcal{D}_{\mathcal{H}}(Q_n, P) \triangleq \sup_{h \in \mathcal{H}} |\mathbb{E}_{Q_n}[h(Z)] - \mathbb{E}_P[h(X)]|, \quad (3.9)$$

where, in case of a weighted sample, $\mathbb{E}_{Q_n}[h(Z)] = \sum_{i=1}^n q_i h(x_i)$. If the class of test functions $\mathcal{H}$ is large enough, for any sequence of probability measures $(Q_n)_{n \geq 1}$, the convergence of $\mathcal{D}_{\mathcal{H}}(Q_n, P)$ to zero implies that Q_n converges weakly to P . One advantage of using this formulation is that we can recover different metrics in the form

(3.9) by selecting $\mathcal{H}$ appropriately (Gorham & Mackey, 2015); for example, using $\mathcal{H} \triangleq \{h : \mathcal{X} \rightarrow \mathbb{R} \mid \sup_{x \in \mathcal{X}} |h(x)| \leq 1\}$, the measure in (3.9) becomes the *total variation distance*, while using the class of Lipschitz-continuous functions $\mathcal{H} \triangleq \{h : \mathcal{X} \rightarrow \mathbb{R} \mid \sup_{x \neq y \in \mathcal{X}} \frac{\|h(x) - h(y)\|_2}{\|x - y\|_2} \leq 1\}$ recovers the Wasserstein distance. Note that, typically, finding the exact solution in (3.9) is as hard as the original problem since one needs to calculate $\mathbb{E}_P[h(X)]$; however, if we can find a class of functions $\mathcal{H}$ such that

$$\mathbb{E}_P[h(X)] = 0 \quad (3.10)$$

for any function $h \in \mathcal{H}$ and $\mathcal{H}$ is sufficiently rich so that (3.9) is still a meaningful measure of similarity, we can avoid this problem.

3.3.1 Stein Discrepancy

Following Gorham & Mackey (2015), in this section we present Stein discrepancy, which is a special case of (3.9) with the desired property (3.10). Let $\partial\mathcal{X}$ be the boundary of $\mathcal{X}$, $n(x)$ be the outward normal unit vector to the boundary at x , $\|\cdot\|$ be a generic norm on $\mathbb{R}^d$, and $\|\cdot\|^*$ be its dual norm.⁸ Then Gorham & Mackey (2015) defined the *classical Stein set* $\mathcal{F}_{\|\cdot\|}$ as

$$\begin{aligned} \mathcal{F}_{\|\cdot\|} \triangleq & \left\{ f : \mathcal{X} \rightarrow \mathbb{R}^d \left| \sup_{x \neq y \in \mathcal{X}} \max \left\{ \|f(x)\|^*, \|\nabla f(x)\|^*, \frac{\|\nabla f(x) - \nabla f(y)\|^*}{\|x - y\|} \right\} \leq 1 \right. \right\}, \\ & \text{and } \langle f(x), n(x) \rangle = 0, \forall x \in \partial\mathcal{X} \Big\}. \end{aligned} \quad (3.11)$$

Furthermore, the *Langevin Stein operator* for p (a mapping from $\{f : \mathbb{R}^d \rightarrow \mathbb{R}^d\}$ to $\{h : \mathbb{R}^d \rightarrow \mathbb{R}\}$) is defined as

$$(T_p f)(x) \triangleq \frac{1}{p(x)} \langle \mathbf{1}, \nabla(p(x)f(x)) \rangle = \langle \nabla \log(p(x)), f(x) \rangle + \langle \mathbf{1}, \nabla f(x) \rangle, \quad (3.12)$$

where $\mathbf{1}$ denotes the d -dimensional all-one vector (with each component being 1). Note that calculating (3.12) depends only on $\nabla \log p(x)$, so the normalization constant is not

⁸The dual norm $\|\cdot\|^*$ is defined as $\|x\|^* = \sup_{y \in \mathbb{R}^d, \|y\|=1} \langle x, y \rangle$ for any $x \in \mathbb{R}^d$.

necessary to be known (e.g., as in typical applications of MCMC). The next proposition shows that $\mathcal{H} = T_p \mathcal{F}_{\|\cdot\|} \triangleq \{h = T_p f : f \in \mathcal{F}_{\|\cdot\|}\}$ satisfies (3.10).

Proposition 3.1. (Gorham & Mackey, 2015, Prop. 1) *If $\mathbb{E}_P[\|\nabla \log p(X)\|_2] < \infty$, then $\mathbb{E}_P[(T_p f(X))] = 0$ for all $f \in \mathcal{F}_{\|\cdot\|}$.*

Then the Stein discrepancy is defined as

$$\begin{aligned} \mathcal{S}(Q_n, \mathcal{H}) &= \mathcal{S}(Q_n, T_p \mathcal{F}_{\|\cdot\|}) \triangleq \sup_{f \in \mathcal{F}_{\|\cdot\|}} |\mathbb{E}_{Q_n}[(T_p f)(Z)]| \\ &= \sup_{f \in \mathcal{F}_{\|\cdot\|}} |\mathbb{E}_{Q_n}[(T_p f)(Z)] - \mathbb{E}_P[(T_p f)(X)]| \\ &= \mathcal{D}_{T_p \mathcal{F}_{\|\cdot\|}}(Q_n, P). \end{aligned} \quad (3.13)$$

Still, the optimization in (3.13) is intractable since it has infinitely many constraints coming from the definition of the Stein set $\mathcal{F}_{\|\cdot\|}$. Gorham & Mackey (2015) suggested a way to approximate this optimization by solving it over a limited number of points: They imposed the smoothness constraints in the classical Stein set (3.11) only on a finite number of points, and provided upper and lower bounds for this approximation to ensure that $\mathcal{S}$ goes to zero only if $Q_n \Rightarrow P$. The resulting approximation requires solving a linear program which is still computationally expensive (Chwialkowski et al., 2016). Therefore, Chwialkowski et al. (2016); Gorham & Mackey (2017); Liu et al. (2016) suggested a kernelized version of this measure which has the advantage of having a closed form solution. This kernelized version is discussed in the next section.

3.3.2 Kernel Stein Discrepancy

Let $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be a (positive definite) kernel, $\mathcal{K}$ be its associated reproducing kernel Hilbert space (RKHS), $\|\cdot\|_{\mathcal{K}}$ be the norm induced by the inner product in $\mathcal{K}$ (Manton & Amblard, 2015). Then, for $x \in \mathbb{R}^d$ and $g \in \mathcal{K}$, we have $g(x) = \langle g, k(x, \cdot) \rangle$. Given this kernel and an arbitrary norm $\|\cdot\|$ on $\mathbb{R}^d$, Gorham & Mackey (2017) defined

the *kernel Stein set* $\mathcal{F}_{k, \|\cdot\|}$ as a set of vector-valued functions $f = (f_1, \dots, f_d)$ such that

$$\mathcal{F}_{k, \|\cdot\|} \triangleq \left\{ f = (f_1, \dots, f_d) \in \mathcal{K}^d \mid \|v\|^* \leq 1 \text{ for } v_i \triangleq \|f_i\|_{\mathcal{K}}, i \in 1, \dots, d \right\}.^9 \quad (3.14)$$

The following proposition shows that $\mathcal{H} = T_p \mathcal{F}_k$ satisfies (3.10).

Proposition 3.2. (Gorham & Mackey, 2017, Prop. 1) *If both $k(x, y)$ and $\nabla_x \nabla_y k(x, y)$ are continuous, bounded, and $\mathbb{E}_P[\|\nabla_x \log p(X)\|_2] < \infty$, then $\mathbb{E}_P[(T_p f(X))] = 0$ for all $f \in \mathcal{F}_k$.*

By replacing the generic norm $\|\cdot\|$ with $\|\cdot\|_2$, one can easily recover the class of test functions used by Chwialkowski et al. (2016) and Liu et al. (2016). Given the Langevin Stein operator (3.12) and the kernel Stein set (3.14), the *kernel Stein discrepancy* (KSD) is defined as $\mathcal{S}(Q_n, T_p \mathcal{F}_k)$. The next proposition shows that for any norm $\|\cdot\|$ there is a “closed form solution” for the KSD.

Proposition 3.3. (Gorham & Mackey, 2017, Prop. 2) *Suppose $k(\cdot, \cdot)$ is differentiable and $\nabla_x \nabla_y k(x, y)$ is continuous. Define $k_p^i : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ for $i \in [d]$ as*

$$\begin{aligned} k_p^i(x, x') &\triangleq s_{p,i}(x)^\top k(x, x') s_{p,i}(x') + s_{p,i}(x)^\top \nabla_{x'_i} k(x, x') \\ &\quad + s_{p,i}(x')^\top \nabla_{x_i} k(x, x') + \nabla_{x_i} \nabla_{x'_i} k(x, x'), \end{aligned}$$

where $s_{p,i}(x) \triangleq \nabla_{x_i} \log p(x)$. Also, let μ be a distribution over $\mathcal{X}$. If $\sum_{i=1}^d \mathbb{E}_\mu[k_p^i(X, X)^{\frac{1}{2}}] < \infty$, then $\mathcal{S}(\mu, T_p \mathcal{F}_k) = \|w\|$ where $w_i \triangleq \sqrt{\mathbb{E}_{\mu \times \mu}[k_p^i(X, X')]}$ and X, X' are independent samples from μ .

The following proposition shows that all KSDs with different norms $\|\cdot\|$ are equivalent in a strong sense. Therefore, later on, the L_2 norm $\|\cdot\|_2$ is used instead of a generic norm.

Proposition 3.4 (Kernel Stein set equivalence). (Gorham & Mackey, 2017, Prop. 3) *Under the assumptions of Proposition 3.3, for a generic norm $\|\cdot\|$ on $\mathbb{R}^d$, there are constants $c_d, c'_d > 0$ depending only on d and $\|\cdot\|$ such that $c_d \mathcal{S}(\mu, T_p \mathcal{F}_{k, \|\cdot\|}) \leq \mathcal{S}(\mu, T_p \mathcal{F}_{k, \|\cdot\|_2}) \leq c'_d \mathcal{S}(\mu, T_p \mathcal{F}_{k, \|\cdot\|})$.*

⁹ To reduce clutter, later on, we drop the norm subindex from $\mathcal{F}_{k, \|\cdot\|}$ and only write $\mathcal{F}_k$ when the norm is clear from the context.

By substituting μ with our previously used discrete distribution, Q_n , we get

$$\mathcal{S}(Q_n, T_p \mathcal{F}_k) = \sqrt{\sum_{j=1}^d \sum_{i, i'=1}^n q_n(x_i) k_p^j(x_i, x_{i'}) q_n(x_{i'})}. \quad (3.15)$$

In what follows, we provide upper and lower bounds for the convergence of the KSD to understand under what conditions $\mathcal{S}_{\mathcal{F}_{k, \|\cdot\|_2}}(\mu_m, P) \rightarrow 0$ implies $\mu_m \Rightarrow P$. We start with defining the family of *distantly dissipative* distributions $\mathcal{P}$. According to Gorham & Mackey (2017), a continuous distribution with density p is distantly dissipative if

$$\kappa_0(p) \triangleq \lim_{r \rightarrow \infty} \inf \kappa(r) > 0$$

where for any $x, y \in \mathcal{X}$,

$$\kappa(r) = \inf \left\{ -2 \frac{\langle s_p(x) - s_p(y), x - y \rangle}{\|x - y\|_2^2} \mid \|x - y\|_2 = r \right\}.$$

A sequence of probability measures $(\mu_m)_{m \geq 1}$ is uniformly tight if for every $\epsilon > 0$, there is a finite number $R(\epsilon)$ such that $\limsup_m \mu_m(\{x \in \mathbb{R}^d \mid \|x\|_2 > R(\epsilon)\}) \leq \epsilon$ (Gorham & Mackey, 2017). Given this definition, we can establish the convergence bounds as follows:

Theorem 3.12 (KSD detects tight non-convergence, Gorham & Mackey 2017, Theorem 7). *Suppose $P \in \mathcal{P}$ and $k(x, y) = \Phi(x - y)$ where Φ is twice continuously differentiable with a non-vanishing generalized Fourier transform.¹⁰ If $(\mu_m)_{m \geq 1}$ is uniformly tight, then $\mathcal{S}(\mu_m, T_p \mathcal{F}_k) \rightarrow 0$ only if $\mu_m \Rightarrow P$.*

Theorem 3.13 (KSD detects convergence, Gorham & Mackey 2017, Prop. 9). *If $\nabla_x^l \nabla_y^l k(x, y)$ is continuous and uniformly bounded for $l \in \{0, \dots, 2\}$, $\nabla_x \log p(x)$ is Lipschitz, and $\mathbb{E}_P[\|\nabla_x \log p(X)\|_2^2] < \infty$, then $\mathcal{S}_{\mathcal{F}_{k, \|\cdot\|_2}}(\mu_m, P) \rightarrow 0$ whenever the Wasserstein distance¹¹ $d_W(\mu_m, P) \rightarrow 0$.*

It is known that the convergence of the Wasserstein distance to zero implies weak convergence (i.e., the Wasserstein distance induces a stronger form of convergence than

¹⁰Define $\hat{\Phi}$ to be the generalized Fourier transform of Φ , then $\hat{\Phi}$ is non-vanishing if $F(t) \triangleq \sup_{\|\omega\|_\infty \leq t} \hat{\Phi}(\omega)^{-1}$ is finite for all $t > 0$.

¹¹The Wasserstein distance is defined in Section 3.3.

weak convergence). So, under the conditions of Theorem 3.12 and Theorem 3.13, such as for unimodal distributions P with log-concave densities, the KSD measure goes to 0 if and only if Q_n converges weakly to P .

The only remaining question is how to choose the kernel function. Gorham & Mackey (2017) discussed that common kernel functions such as the Gaussian kernel ($e^{-\frac{1}{2}\|x-y\|_2^2}$) fail to detect non-convergence for $d \geq 3$ (i.e., KSD can go to 0 without $Q_n \Rightarrow P$) since KSDs based on these kernels do not enforce the required uniform tightness (Gorham & Mackey, 2017, Theorem 6). Gorham & Mackey (2017) recommended the inverse multiquadric (IMQ) kernel, defined as

$$k(x, y) = (c^2 + \frac{1}{h}\|x - y\|_2^2)^\beta, \quad (3.16)$$

with some $\beta \in (-1, 0)$ and $h, c > 0$. They associated the success of this kernel to its slow decay rate (for large $\|x - y\|$). Theorem 8 of Gorham & Mackey (2017) shows that using this kernel, KSD detects non-convergence KSD, $\mathcal{S}(Q_n, P) \rightarrow 0$ only if $Q_n \Rightarrow P$.

However, despite of the above nice theoretical guarantees, one might not be able to detect convergence based on KSD for practical sample sizes, especially when the target distribution is multimodal and the modes are well-separated: Gorham et al. (2016) demonstrated that for a one-dimensional Gaussian mixture target distribution (with two components), for practical sample sizes, the KSD measure fails to distinguish between two sets of samples, one drawn independently from one mode and the other drawn independently from the whole target distribution. Furthermore, KSD requires even more samples to distinguish between the two cases as the modes' distance increases (see Section 6.1 of Gorham et al. (2016) for more details). Another issue is that the complexity of computing the KSD for an empirical distribution is quadratic in the sample size, which quickly becomes infeasible as the sample size grows.

An interesting property of the KSD measure is that it is convex: if $Q^1, \dots, Q^M$ are distributions and $Q = \sum_{i=1}^M w_i Q^i$ for nonnegative weights w_i satisfying $\sum_{i=1}^M w_i = 1$,

we have

$$\begin{aligned} \mathcal{S}(Q) &= \sup_{f \in \mathcal{F}_{k, \|\cdot\|}} |\mathbb{E}_Q [(T_p f)(Z)]| = \sup_{f \in \mathcal{F}_{k, \|\cdot\|}} \left| \sum_{i=1}^M w_i \mathbb{E}_{Q^i} [(T_p f)(Z)] \right| \\ &\leq \sup_{f \in \mathcal{F}_{k, \|\cdot\|}} \sum_{i=1}^M w_i |\mathbb{E}_{Q^i} [(T_p f)(Z)]| \leq \sum_{i=1}^M w_i \sup_{f \in \mathcal{F}_{k, \|\cdot\|}} |\mathbb{E}_{Q^i} [(T_p f)(Z)]| = \sum_{i=1}^M w_i \mathcal{S}(Q^i), \end{aligned}$$

where $\mathcal{S}(Q)$ is used instead of $\mathcal{S}_{\mathcal{F}_k}(Q, T_p \mathcal{F}_k)$ to reduce clutter.

3.4 Background: Multi-Armed Bandit Problems

It has already been discussed in Section 3.1.1 that our goal is to design an adaptive MCMC algorithm which sequentially allocates the computational budget among parallel MCMC samplers to have a fast mixing sampler over the whole state space. In order to design such a sequential sampler, we make use of some algorithms developed in the literature for multi-armed bandit problems. In what follows, we provide a brief introduction to the latter topic.

Making decisions in the presence of uncertainty is one of the many challenges that we face on a daily basis, and, naturally, it is important in industrial and scientific problems too. Which drug should be given to a patient? Which website design will attract more donors for a charity campaign? Which restaurant to choose when eating out in a new city? In all of these problems, our goal is to maximize the long term payoffs of our decisions, while the outcome of our decisions is not known in advance to us.

Multi-armed bandit problems are basic examples of sequential decision making tasks when the payoff (consequence) of each decision is not known in advance (Bubeck & Cesa-Bianchi, 2012). In a multi-armed bandit problem, an agent interacts with an environment over multiple time steps. In each time step the agent takes an action, and receives some observable payoff. The goal of the agent is to maximize the total payoff obtained over a sequence of decisions. The agent has two natural options: choosing an action that provided high payoffs in the past (exploiting his knowledge), or trying other actions with the hope that they might provide higher payoffs in the future (this

is called exploration). In order to maximize the total payoff, the agent should find a balance between exploration and exploitation, which is a fundamental tradeoff in sequential decision making.

There are two main types of multi-armed bandit problems: stochastic and adversarial. This categorization is based on the nature of the payoffs and there are different strategies to play each of them (Bubeck & Cesa-Bianchi, 2012). In this thesis we are mainly interested in the stochastic setting.

Formally, let $M \geq 2$ be the number of actions, and $\{X_{i,t}\}_{t=1}^n$ be a sequence of payoffs for each action $i \in \{1, \dots, M\}$. The agent at each time step $t \in \{1, \dots, n\}$, where n is the time horizon, takes an action, I_t , and receives the payoff, $X_{I_t,t}$. The goal of the agent is to maximize the sum of these payoffs $\sum_{t=1}^n X_t$. Equivalently, the performance of the agent is evaluated by the relative loss it suffers compared to choosing the best arm in hindsight, called a regret: Then the regret is defined as

$$R_n \triangleq \max_{i \in [M]} \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t}. \quad (3.17)$$

Note that both the rewards and the forecaster's actions can be stochastic. Instead of directly dealing with the random quantity R_n , in the literature the following two types of expected regret have been considered (Bubeck & Cesa-Bianchi, 2012): (i) the expected regret

$$\mathbb{E}[R_n] = \mathbb{E} \left[\max_{i \in [K]} \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right];$$

(ii) the pseudo-regret

$$\bar{R}_n \triangleq \max_{i \in [K]} \mathbb{E} \left[\sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right].$$

From the above, we can immediately see that $\bar{R}_n \leq \mathbb{E}[R_n]$, since the expectation is taken over the maximum of $X_{i,t} - \sum_{t=1}^n X_{I_t,t}$. Indeed, the pseudo-regret is a weaker definition of regret, since the forecaster is compared with an action that is optimal in expectation; while, for expected regret, the forecaster is compared with the action that is optimal on the actual realization of the payoffs (Bubeck & Cesa-Bianchi, 2012).

Algorithm 7 UCB1 (Auer et al., 2002)**Initialization:** for $t \leq K$ play each arm once.**for** $t = K + 1, K + 2, \dots, n$ **do**Play arm i that maximizes

$$\bar{\mu}_{i,t-1} + \sqrt{\frac{2 \log t}{T_i(t-1)}}. \quad (3.19)$$

Set $T_i(t) = T_i(t-1) + 1$ and $T_j(t) = T_j(t-1)$ for $j \neq i$.Observer $X_{i,T_i(t)}$, and compute $\bar{\mu}_{i,t} = (1 - \frac{1}{T_i(t)})\bar{\mu}_{i,t-1} + \frac{X_{i,T_i(t)}}{T_i(t)}$.**end for**

In the stochastic multi-armed bandit setting, we assume that the payoffs of each arm i , $\{X_{i,t}\}$, are drawn independently from an unknown distribution ν_i , $i \in [M]$. Let $T_k(t) = \sum_{s=1}^t \mathbb{I}_{\{I_s=k\}}$ be the number of times action i is chosen by the forecaster up to time t , μ_i be the mean of ν_i , and $\mu^* = \max_{i \in [M]} \mu_i$. Then the pseudo-regret becomes $\bar{R}_n = \sum_{k=1}^M \Delta_i \mathbb{E}[T_k(n)]$, where $\Delta_i = \mu^* - \mu_k$. The reason to choose pseudo-regret is that in a stochastic problem it makes more sense to compare the performance with the optimal action in expectation instead of comparing it with the optimal action over the realized payoffs (Bubeck & Cesa-Bianchi, 2012).

A standard algorithm for this problem is the the UCB1 algorithm of Auer et al. (2002), shown in Algorithm 7. UCB1 uses the concept of *optimism in the face of uncertainty* to do exploration and exploitation simultaneously (Bubeck & Cesa-Bianchi, 2012). The algorithm computes a confidence interval for the expected payoff of each action, and, at each step, chooses the arm that has the highest upper confidence, see (3.19) in Algorithm 7. Assume $X_{k,t} \in [0, 1]$. Auer et al. (2002) showed that the pseudo-regret of UCB1 is bounded by

$$\bar{R}_n \leq \sum_{k: \Delta_k > 0} \frac{8 \log n}{\Delta_k} + \left(1 + \frac{\pi^2}{3}\right) \Delta_k. \quad (3.18)$$

3.5 Sequential Selection of Samplers

In previous sections, we reviewed MCMC methods, introduced our quality measure of interest, and discussed the stochastic multi-armed bandit problem and its relevance to the problem at hand. With this background, we are ready to present our new adaptive sampling algorithms. Thus, in this section, we present several strategies to select from a pool of MCMC samplers in a sequential manner. In all of our algorithms the selection of the sampler to be used next depends on the quality of the samples generated by the different samplers, where the quality will be measured by the KSD measure (or its approximations). Formally, assume we have access to M MCMC samplers (e.g., multiple sampling methods and/or multiple instances of the same sampling algorithm with different parameters, such as starting point or step size), and denote the set of samplers by $[M]$. At every step of the algorithm, we select one of the samplers and use it to produce the next sample (or the next batch of samples).

3.5.1 Mixing Samplers

First we consider the case when each sampler is asymptotically unbiased (or consistent), that is, it generates samples with an empirical distribution converging weakly to the target P almost surely (this is usually satisfied for any standard MCMC sampler when P is unimodal). Our task is to sequentially allocate calls among the M samplers to minimize the Stein discrepancy of the set of samples we collect. In order to do so, we design an algorithm which gives preference to samplers where the convergence is faster. This setup is similar to the one considered by Neufeld et al. (2014), who designed sequential sampling strategies for MC samplers generating independent and identically distributed (i.i.d.) samples, by selecting samplers with smaller variances based on multi-armed bandit algorithms. In this section we generalize their method to MCMC samplers, aiming to minimize the total KSD measure (cf. Eq. 3.15) of the sample instead of the variance (recall that for practical sample sizes—before convergence—the variance of the sample is not a good measure of quality for MCMC samplers). However, as mentioned in Section 3.3.2, computing the Stein discrepancy even for relatively small sample sizes is computationally infeasible (also note that any computation we

spend on selecting samplers could also be used for sampling). Therefore, we are going to approximate the KSD measure as the average KSD over smaller blocks of samples.

For a sampler with total sampling budget n , we break the sampling process into T rounds: At each round the sampler takes a batch of samples of size $n_b = n/T$. Let S_t be the KSD measure of samples from the t th round; then we will approximate $\tilde{S}_n$, the KSD of the full sample of size n with the average $(1/T) \sum_{t=1}^T S_t$. We will call this the *block-diagonal approximation*, as it corresponds to a block-diagonal approximation of the kernel matrix in (3.15). To quantify the accuracy of the approximation, we assume that there exists a function $g(t, n_b)$ such that $\lim_{\frac{n_b}{t} \rightarrow \infty} g(t, n_b) = 0$ and $\frac{1}{t} \sum_{b=1}^t \mathbb{E}[S_b] - \mathbb{E}[\tilde{S}_{tn_b}] \leq g(t, n_b)$ for all $t = 1, \dots, T$. Using the block-diagonal approximation, we relax our goal to competing with a sampler with the smallest average approximate block-KSD measure $(1/T) \sum_{t=1}^T S_{i,t}$, where $S_{i,t}$ is the KSD measure of the n_b samples generated by sampler i when it is called the t th time. We also assume that this is close to the KSD measure $\tilde{S}_{i,t}$ of a sample of size tn_b obtained from using sampler i for t blocks, that is $\frac{1}{t} \sum_{b=1}^t \mathbb{E}[S_{i,b}] - \mathbb{E}[\tilde{S}_{i,tn_b}] \leq g(t, n_b)$ for $t = 1, \dots, T$. Experimental results presented in Section 3.7.1 indicate that the block-diagonal approximation mostly preserves the ranking of the samplers (as defined by the true KSD measure), hence we pay very little price for the computational advantage we get.

Furthermore, solving this problem is well-suited for any multi-armed bandit algorithm (Bubeck & Cesa-Bianchi, 2012); here we adapt the UCB1 method of Auer et al. (2002), given in Algorithm 8.¹² In short, the algorithm keeps track of an optimistic estimate of the average approximate KSD value for each sampler, and selects a sampler whose performance is estimated to be the best possible (based on high-probability confidence intervals).

If the KSD values $S_{i,1}, \dots, S_{i,T}$ were i.i.d., the standard multi-armed bandit regret bound (Bubeck & Cesa-Bianchi, 2012) would yield $\sum_{t=1}^T \mathbb{E}[S_t] - \min_{i \in [M]} \sum_{t=1}^T \mathbb{E}[S_{i,t}] = O(\log(T))$. Using our assumption and the convexity of the KSD measure (recall

¹²The algorithm assumes that $S_{i,t} \in [0, 1]$, which can be achieved by rescaling the KSD measures. In practice, a reasonable estimate for the range can be obtained from $S_{i,1}$ for each sampler i .

Algorithm 8 KSD-UCB1

```

for  $i \in \{1, \dots, M\}$  do
  - Use sampler  $i$  to generate  $n_b$  samples; observe  $S_{i,1}$ ; set  $\bar{\mu}_{i,1} = S_{i,1}$  and  $T_i(1) = 1$ .
end for
for  $t \in \{M+1, M+2, \dots, n\}$  do
  - Play arm  $i$  that minimizes  $\bar{\mu}_{i,t-1} - \sqrt{\frac{2 \log t}{T_i(t-1)}}$ .
  - Set  $T_i(t) = T_i(t-1) + 1$  and  $T_j(t) = T_j(t-1)$  for  $j \neq i$ .
  - Observe  $S_{i,T_i(t)}$ , and compute  $\bar{\mu}_{i,t} = (1 - \frac{1}{T_i(t)})\bar{\mu}_{i,t-1} + \frac{S_{i,T_i(t)}}{T_i(t)}$ .
end for

```

Eq. 3.17), this would imply that after T rounds of sampling,

$$\mathbb{E}[\tilde{S}_{Tn_b}] - \min_i \mathbb{E}[\tilde{S}_{i,Tn_b}] \leq \frac{1}{T} \left(\sum_{t=1}^T \mathbb{E}[S_t] - \min_{i \in [M]} \sum_{t=1}^T \mathbb{E}[S_{i,t}] \right) + 2g(T, n_b) \approx \frac{O(\log T)}{T} + 2g(T, n_b).$$

This shows that increasing T and n_b , the performance of KSD-UCB1 would be close to using the best sampler. However, in our case, the $S_{i,t}$ are not i.i.d. Assuming that the samplers mix (which is reasonable for a single mode distribution), the $S_{i,t}$ are getting closer and closer to be sampled i.i.d. as n_b increases. Also, as mentioned above, the effect of the block-diagonal approximation (and hence that of $g(T, n_b)$) is small in practice (see also Section 3.7.1). The exact parametrization of KSD-UCB1 corresponds to the assumption that the block-KSD values are in $[0, 1]$. While this is not true in general, in practice we estimate the maximum KSD value from samples and normalize the values accordingly, before feeding them to the KSD-UCB1 algorithm.

3.5.2 Locally Mixing Samplers

In practice, if the target distribution is multimodal and the modes are far from each other, MCMC methods often get stuck in some of the modes and fail to explore all the regions where P is supported; while eventually all asymptotically consistent methods reach each mode, this may not happen after any practically reasonable time. To model this situation, we assume that the support of P is partitioned into sets $A_1, \dots, A_K$ with $P(A_j) > 0$ for all $j \in [K]$ (the A_j are pairwise disjoint and their union is the support of P) such that the empirical distribution of the samples generated by sampler $i \in [M]$ converges weakly to $P(\cdot|A_j)$ for some $j \in [K]$, where $P(\cdot|A)$ is the conditional

distribution of P over A . We will refer to the sets A_j as *regions*, and a sampler satisfying the above condition a *locally mixing* sampler.

For simplicity we first consider the case where there is one sampler in each region A_i (consequently $M = K$). This setup is similar to stratified sampling, which is a variance reduction technique for MC. The idea is to partition the domain into non-overlapping regions (a.k.a. strata), draw samples from each region, and combine the final samples to estimate $\mu = \mathbb{E}[f(X)]$ (Owen, 2013). The problem in stratified sampling is to find the optimal number of samples that need to be taken from each stratum in order to reduce the variance of Monte Carlo integration error. Carpentier et al. (2015) showed that an optimal strategy for minimizing the mean squared error (MSE) is to sample each stratum $n_i = \frac{\sigma_i P(A_i)}{\sum_{i=1}^K \sigma_i P(A_i)} n$ times, where σ_i is the conditional standard deviation of $f(X)$ given that X falls into the i th region. They also gave an adaptive sampling algorithm, based on a reduction to multi-armed bandit problems, for the case when the standard deviations are unknown.

One can immediately see that the setup we consider in this subsection is very similar to the last case with the important differences that our samplers are not i.i.d., and we do not minimize the squared error but the KSD measure. Denoting the distribution of samples from region A_i by $Q_{n,i}$ after taking n samples in total, let w_i denote the weight of sampler i generating these samples (recall that here, by assumption, we have one sampler in each region). Then our total weighted sample distribution becomes

$$Q_n = \sum_{i=1}^M w_i Q_{n,i}. \quad (3.20)$$

Since according to our assumptions, $Q_{n,i}$ converges weakly to $P(A_i)$ almost surely, for every i , we need to have $w_i \rightarrow P(A_i)$ for all $i \in [M]$ to ensure that Q_n converges to P weakly (we will refer to this as the w_i being asymptotically consistent). A procedure for estimating w_i this way will be given in the next section. Assuming for a moment that $w_i = P(A_i)$, using the convexity of the KSD measure (see Eq. 3.17), we have

$$\tilde{S}_n = \mathcal{S}(Q_n) \leq \sum_{i=1}^M w_i \mathcal{S}(Q_{n,i}) = \sum_{i=1}^M w_i \tilde{S}_{i,n_i}, \quad (3.21)$$

Algorithm 9 KSD-UCB1-M

Given: (Unnormalized) density $\hat{p}$; partition of the domain $A_1, \dots, A_K$; M samplers in K classes: samplers in $\mathcal{A}_i$ sample from $p(\cdot|A_i)$ for all $i \in [K]$, $\cup_{i \in [K]} \mathcal{A}_i = [M]$; total number of rounds: T ; batch size: n_b .

Initialize: Draw a batch of samples from each sampler i , observe $S_{i,1}$ and set $\hat{\mu}_{i,1} = S_{i,1}$ and $T_i(1) = 1$ for all $i \in [M]$.

for $t \in \{M+1, \dots, T\}$ **do**

- Select a region I_t .

- Draw a batch of samples from arm $i_t = \arg \min_{i \in A_{I_t}} \left(\mu_{i,t-1} - \sqrt{\frac{2 \log t}{T_i(t-1)}} \right)$.

- Set $T_{i_t}(t) = T_{i_t}(t-1) + 1$ and $T_j(t) = T_j(t-1)$ for $j \neq i_t$.

- Observe $S_{i_t, T_{i_t}(t)}$, and compute $\bar{\mu}_{i_t, t} = \left(1 - \frac{1}{T_{i_t}(t)}\right) \bar{\mu}_{i_t, t-1} + \frac{S_{i_t, T_{i_t}(t)}}{T_{i_t}(t)}$.

end for

- Reweight the samples of each mode proportional to its weight, $\frac{w_k}{T n_b}$.

where, as before, $\tilde{S}_n$ and $\tilde{S}_{i, w_i n}$ denote the KSD measure of the whole sample and, respectively, that of sampler i (with number of samples n_i obtained by sampler i). Based on this inequality, we could aim for minimizing $\sum_{i=1}^M w_i \tilde{S}_{i, n_i}$ and use the ideas from adaptive stratified sampling (Carpentier et al., 2015); however, multiple challenges preclude us from doing this: (i) w_i is not known in advance; (ii) the stratified sampling algorithm is based on the known concentration of the variance, but we do not know how fast $\tilde{S}_{i, n_i}$ approaches zero (as a function of $n_i = w_i n$); and (iii) the computational complexity of calculating $\tilde{S}_{i, n_i}$ is $O(n_i^2)$. To handle (i), we will address the problem of estimating w_i in Section 3.5.3. For (ii), a conservative approach is to uniformly minimize $w_i \tilde{S}_{i, n_i}$, hence selecting sampler i for which this quantity is the largest. For (iii), we will again use a block-diagonal approximation to $\tilde{S}_{i, n_i}$, which causes problems with (ii), since the estimate does not converge to 0 for a fixed block size. In Section 3.7.3 we present experiments with different strategies under different setups, but the strategies considered seem to perform rather similarly. The simplest strategy considered is to select regions uniformly at random. Another approach is to minimize the maximum $\tilde{S}_{i, n_i}$, that is, choosing $i = \arg \max \tilde{S}_{i, n_i}$, which does not require the knowledge of the weights w_i , while we also test strategies selecting regions based on their estimated weights or, similarly to stratified sampling, based on their estimated variance. Although quite different, the strategies considered seem to perform rather similarly in various settings.

Unfortunately, we cannot guarantee that we start samplers in such a way that we

have a single sampler for each region. If we know which sampler belongs to which region, we can combine the region selection strategies described above with the bandit method described in Section 3.5.1 in a straightforward way: in each region, run an instance of KSD-UCB1 over the samplers exploring this region, and use it as a locally mixing sampler above. We will refer to this sampling algorithm as KSD-UCB1-M (M stands for Multiple regions); the pseudocode of the algorithm is given in Algorithm 9. Clearly, since the bandit algorithms sample each arm infinitely often, we have the following consistency result:

Under the local mixing assumption made at the beginning of this section, the final weighted sample (3.20) obtained by KSD-UCB1-M is asymptotically unbiased as long as the weight estimates w_i are asymptotically unbiased.

Thus, we need to find some asymptotically unbiased estimates of the probabilities of the regions A_i .

3.5.3 Weight Estimation

In this section we consider the problem of finding the weights w_i in (3.20). As discussed after the equation, this amounts to finding the probability of the region the samples cover, which is again challenging since we have access only to an unnormalized density. This problem is faced by every algorithm which tries to speed up MCMC methods by running parallel chains. As an example, in big-data scenarios it is common to split the data and try to run an MCMC sampler on each subset to combine the final samples. To our knowledge, most work in the literature solves this problem by estimating the density of each batch of samples separately (Angelino et al., 2016; Nemeth & Sherlock, 2018), using typically either a Gaussian approximation (Scott et al., 2016) or some kernel-density estimation method (Neiswanger et al., 2014). According to Nemeth & Sherlock (2018), the first approach works well in practice, in spite of not having any theory; while the second, kernel based estimation scales poorly with the dimension d .

Here we take a different approach and rather than estimating the density of the sample batches, we directly estimate the probabilities $P(A_i)$ via Rényi entropy.

Formally, suppose the domain of P is partitioned into non-overlapping regions $A_1, \dots, A_K$, and from each region A_i we have a set of samples $X^{(A_i)}$. The Rényi entropy of order $\alpha \neq 1$ for a density p is defined as

$$R_\alpha(p) = \frac{1}{1-\alpha} \log \int_{\mathbb{R}^d} p^\alpha(x) dx.$$

The conditional density of P restricted to a set A is denoted as $p(x|A) = \frac{p(x)}{P(A)} \mathbb{I}_{\{x \in A\}}$, and its Rényi entropy is

$$R_\alpha(p|A) = \frac{1}{1-\alpha} \log \int_A \left(\frac{p(x)}{P(A)} \right)^\alpha dx.$$

From this definition it trivially follows that

$$\log P(A) = R_\alpha(p|A) - \frac{1}{1-\alpha} \log \mathbb{E} [p(X)^{\alpha-1} | X \in A].$$

In our case, instead of p we only have access to $\hat{p} = cp$ for some $c > 0$. Replacing p with $\hat{p}$ in the integral, we obtain

$$\log(P(A)c) = R_\alpha(p|A) - \frac{1}{1-\alpha} \log \mathbb{E} [\hat{p}(X)^{\alpha-1} | X \in A].$$

Thus, we can estimate $P(A)$ (or, more precisely, $P(A)c$) by estimating the two terms on the right-hand side of the above equation.

Given a sample $X^{(A)}$ taken i.i.d. from $p(\cdot|A)$, the second term can be estimated by the empirical average

$$\hat{B}_\alpha(X^{(A)}) = \frac{1}{|X^{(A)}|} \sum_{x \in X^{(A)}} \hat{p}(x)^{\alpha-1},$$

while for the first term we can use a graph-based estimate (Hero & Michel, 1999). In particular, we are going to use the estimator $\hat{R}_\alpha(X^{(A)})$ of Pál et al. (2010), which is based on generalized k -nearest neighbor graphs of the sample $X^{(A)}$, and it converges to $R_\alpha(p|A)$ almost surely for any $\alpha \in (0, 1)$ as the sample size grows to infinity. Thus, we obtain that $\beta(X^{(A)}) = \hat{R}_\alpha(X^{(A)}) - \hat{B}_\alpha(X^{(A)})$ is an asymptotically unbiased estimate

of $\log(P(A)c)$. Therefore, given a fixed partition $A_1, \dots, A_K$,

$$P(A_i) \approx w_i = e^{\beta(A_i)} / \sum_{j \in [K]} e^{\beta(A_j)} . \quad (3.22)$$

More precisely, if the minimum number of samples in the regions is m , then $P(A_i) = \lim_{m \rightarrow \infty} w_i$.

It is also possible to determine the rate of convergence in the above: Theorem 2 of Pál et al. (2010) shows that with a proper parametrization of the estimator $\hat{R}_\alpha(X^{(A)})$, its (additive) error after $m = |X^{(A)}|$ samples is of order $m^{-1/(d+\gamma)}$ with high probability for an arbitrary choice of $\gamma > 0$ (here we suppressed the dependence on the probability of error in the notation). On the other hand, the error of $\hat{B}_\alpha(X^{(A)})$ is of order $O(m^{-1/2})$ by standard concentration inequalities if p is bounded (see, e.g., Boucheron et al., 2013). This implies that $|\beta(X^{(A)}) - \log(P(A)c)| = O(m^{-1/(d+\gamma)})$, and so $P(A_i)$ can be estimated with a multiplicative error of $O(\exp(\text{const} \cdot m^{-1/(d+\gamma)}))$ where m now denotes the minimum number of samples over the partition cells $A_1, \dots, A_K$. In other words, $|\log P(A_i) - \log w_i| = O(m^{-1/(d+\gamma)})$, and using $e^x = 1 + O(x)$ as $x \rightarrow 0$, we also have $|P(A_i) - w_i| = O(m^{-1/(d+\gamma)})$ as $m \rightarrow \infty$. Note that although the error of our estimator scales quite unfavorably in the dimension d , in practice it seems to work well even for moderately large dimensions (around 20). Furthermore, in the experiments we choose α to be close to one to control the variance of the second term on the right-hand side of (??); see Section 3.7 for details.

3.6 The Final Algorithm

So far we have discussed how to solve our problem if we have either local mixing or all the samplers mix. While the latter is the case asymptotically for all the MCMC samplers used in practice, the mixing may be too slow to be observed for any practical number of samples. On the other hand, the problem does not simplify to the local mixing scenario, since—even if well-separated regions are actually present—the chains often jump from one region to another even if they do not cover the whole domain.

To be able to adapt our KSD-UCB1-M algorithm, we need to group together the samplers covering the same region (even though what a region is is not clearly defined in this scenario). The problem is especially hard since the grouping of the samplers is non-stationary, and we should also be able to track when a sampler leaves or joins a region (equivalently, group). Furthermore, if the groups are too large, we do not explore the whole domain, while if they are too small, we waste resources by running multiple samplers for the same region.

To solve this issue, we propose a simple heuristic to identify samplers that are close together: In each round of the sampling, we take all the samples from the last batch of each sampler, and for each sample point we look at its N nearest neighbors. Then we find the grouping where any two samplers are grouped together that have points which are nearest neighbors of each other (this can easily be done recursively). By this simple heuristic, we can group the samplers that are close together at each round. Note that here we do not make any assumption regarding the number of the regions (e.g., well-separated modes) of the distribution. By having all the samples, the algorithm can easily identify multiple regions by running a clustering algorithm. The final step of the algorithm is to determine the correct weight for each sample point. Recall that in the locally mixing case we weighted the empirical distribution of each region by their estimated probability (cf. Eq. 3.20). Here, since we do not have these regions, in the end we assign the samples into M clusters using k -means clustering, and weight the empirical measure within each cluster by the estimated probability of the cluster (using Eq. 3.22). Algorithm 10 shows the whole procedure, called KSD-MCMC with reweighting (KSD-MCMC-WR).

3.7 Experiments

In this section we empirically evaluate our choices in the algorithm design process as well as compare the performance of our final algorithm with state-of-the-art methods from the literature on several synthetic problems.

We use three different base sampling methods: Metropolis-Hastings (MH), the Metropolis-adjusted Langevin algorithm (MALA) of Roberts & Rosenthal (2002) and

Algorithm 10 KSD-MCMC-WR

Given: Distribution $p(x)$; M samplers; total number of rounds T ; batch size n_b ; number N of nearest neighbors for clustering; the order α of the Rényi entropy.

Initialize: For each $i \in [M]$, draw n_b samples from the i th sampler with random initialization; compute $S_{i,1}$ and set $\bar{\mu}_{i,1} = S_{i,1}$ and $T_i(1) = 1$.

for $t \in \{M+1, \dots, T\}$ **do**

- Cluster the samplers by clustering the samples from their last batches:
 - Initially, the last batch of samples from each sampler forms a cluster.
 - Merge two clusters if any point of one cluster has a point from the other cluster among its N nearest neighbors.
 - Find the number of clusters n_c .
 - Define $\mathcal{A}_i \subset [M]$ for $i \in [n_c]$ as the set of samplers belonging to cluster i ($\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for $i \neq j$).
- Choose a cluster I_t (e.g., uniformly at random).
- Draw a batch of samples of size n_b from sampler $i_t = \arg \min_{i \in I_t} \left(\bar{\mu}_{i, T_i(t-1)} - \sqrt{\frac{2 \log t}{T_i(t-1)}} \right)$.
- Set $T_{i_t}(t) = T_{i_t}(t-1) + 1$ and $T_j(t) = T_j(t-1)$ for $j \neq i_t$.
- Observe $S_{i_t, T_{i_t}(t)}$, and compute $\bar{\mu}_{i_t, T_{i_t}(t)} = \left(1 - \frac{1}{T_{i_t}(t)}\right) \bar{\mu}_{i_t, T_{i_t}(t-1)} + \frac{S_{i_t, T_{i_t}(t)}}{T_{i_t}(t)}$.

end for

- Cluster all the samples into M clusters by k-means clustering; for each cluster calculate its estimated probability w_i using (3.22) and output the reweighted samples with weight w_i/n_i where n_i is the total number of samples in cluster $i \in [M]$.

the no-U-turn sampler (NUTS) of Hoffman & Gelman (2014). For the latter, which is a state-of-the-art sampling method (Liu & Lee, 2016; Wang et al., 2013), we used the implementation provided in the pymc3 package (Salvatier et al., 2016) which, on top of the original NUTS, also includes several modifications recommended in the literature for boosting its performance (see the pymc3 package website for more details). Beside the initial point, MH and MALA have a single step size parameter (for MALA the preconditioning matrix is always the identity), while for NUTS we used the default setting (in our experiments the performance of NUTS was insensitive to the choice of its parameters). In our experiments, when a MH or MALA sampler is chosen randomly, the step size is selected uniformly at random from $[0.1, 5]$ (the initial points are selected uniformly at random “not too far” from the modes of the underlying distribution).

We present one set of experiments for each component of our algorithm. Throughout this section, to compute the KSD measure, we use the inverse multiquadratic kernel $k(x, y) = (1 + \|x - y\|_2^2/h)^\gamma$ suggested by Gorham & Mackey (2017) with $\gamma = -0.5$ and $h = 1$ unless otherwise stated (experiments in Section 3.7.1 and 3.7.6 indicate

that the performance of our method is quite insensitive to the choice of h in the range $[0.01, 100]$.

Section 3.7.1 analyzes the effect of using the block-diagonal approximation of the KSD measure. The performance of our bandit-based samplers for unimodal target distributions is considered in Section 3.7.2. Multimodal densities with separated modes and one sampler in each mode are considered in Section 3.7.3, including an empirical analysis of our weight estimation method in Section 3.7.4. Our final algorithm KSD-MCMC-WR is tested in the general setting of a multimodal target distribution with an unknown number of modes in Section 3.7.5, and specifically against parallel MCMC methods in Section 3.7.6. Finally, we consider a realistic task of node localization in a sensor network in Section 3.7.8.

3.7.1 Block-Diagonal Approximation of the KSD Measure

The intuition behind our bandit MCMC method is that we can identify the best sampler among a group of samplers from the average KSD for small batch sizes instead of making the decision based on the KSD computed over a large set of samples. To verify this hypothesis, we checked if the average approximate KSD computed on small blocks preserves the order of two samplers as defined by the true KSD measure. In particular, we compared the average block-KSD measure of two samplers for different block sizes. We drew $n = 100000$ samples from each sampler for a non-isotropic two-dimensional independent Gaussian distribution and computed the average block-KSD measure for these samples with block sizes 10, 25, 50, 100, 250, 500, 2000 (the parameters of the samplers and the distribution were selected randomly¹³). Formally, for each sampler i with block size n_b and total sample size n , we computed the average block-KSD as $\bar{S}_{i,n_b} = \frac{n_b}{n} \sum_{b=1}^{n/n_b} S_{i,b}$, where $S_{i,b}$ is the average block-KSD computed on the b th block of data, and for a given block size n_b , $\operatorname{argmax}_i \bar{S}_{i,n_b}$ is declared to be the better sampler according to the measurements.

Table 3.1 shows the fraction of times the average block-KSD for different block sizes gave the same ordering of the samplers as the ordering obtained for block size 2000

¹³Throughout this section the variance of the components of a Gaussian distribution are selected uniformly from $[0.1, 3]$

Kernel width	Batch Size	10	25	50	100	250	500	2000
$h = 0.01$	MH	80%	84%	86%	94%	96%	96%	100%
$h = 0.01$	MALA	100%	100%	100%	100%	100%	100%	100%
$h = 0.1$	MH	58%	68%	76%	84%	90%	92%	100%
$h = 0.1$	MALA	86%	94%	94%	96%	98%	98%	100%
$h = 1$	MH	93%	95%	94%	94%	94%	96%	100%
$h = 1$	MALA	93%	92%	93%	91%	93%	94%	100%
$h = 10$	MH	92%	92%	92%	92%	94%	96%	100%
$h = 10$	MALA	68%	68%	76%	84%	84%	84%	100%
$h = 100$	MH	83%	85%	85%	85%	85%	89%	100%
$h = 100$	MALA	77%	79%	83%	85%	81%	81%	100%

TABLE 3.1: The fraction of times the samplers are ranked in the same order for different block sizes as for block size 2000 (regarded as the “ground truth”) for different kernel widths h .

(which is treated as the ground truth in this experiment), both for MH and MALA as base samplers. We can observe that the ordering obtained from the average block-KSD measures is most of the time close to the “ground truth”, justifying its use for measuring sample quality. The agreement in most cases is around at least 90%, and rarely gets below 80%. Interestingly, no trend regarding the kernel width h is observable. Scatter plots of average block-KSD differences ($\bar{S}_{1,n_b} - \bar{S}_{2,n_b}$) for $h = 1$ and different block sizes are given in Figure 3.2 for the MH algorithm. These figures show that in cases of incorrect ordering, the average block-KSD values of the samplers are really close for both block sizes, which means the two samplers considered are of approximately the same quality according to both measures (thus, practically it does not matter which of them is used for sampling).

3.7.2 Unimodal Target Distributions with Multiple Samplers

In this section we consider sampling from a standard normal Gaussian distribution, where the samplers have access to the unnormalized density $p(x) = e^{-x^\top x/2}$. The goal of the samplers is to estimate the mean of the distribution with respect to the mean squared error. We consider 5 different MH and, respectively, MALA samplers (with step size parameters 0.1, 0.2, 0.5, 1, 2, respectively) and run our proposed algorithm KSD-UCB1 (Algorithm 8) on top of them. The method is compared with the following baselines and variations:

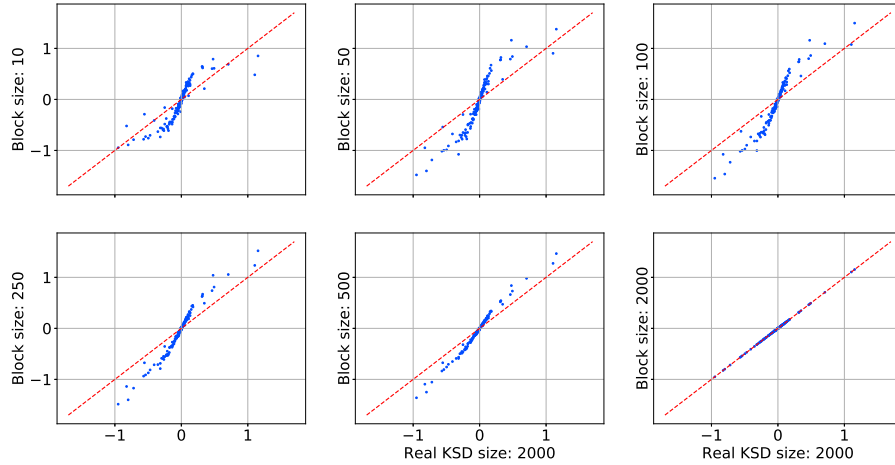


FIGURE 3.2: Block-diagonal approximation of the KSD measure for kernel width $h = 1$.

- **Uniform:** This algorithm distributes the computational budget equally among the samplers, and in the end takes all the samples generated from all samplers with equal weights.
- **KSD-opt:** This method again takes the same number of samples from each sampler (i.e., uses the aforementioned uniform algorithm), but reweights them using the method of Liu & Lee (2016). Indeed, KSD-opt is a reweighting method that can be used with any sets of samples. However, it has been originally proposed to be used on top of the uniform algorithm. This reweighting can be very effective for unimodal distributions or multimodal distributions with really close modes, but it is computationally quite demanding.
- **ε -greedy:** This method is a variant of KSD-UCB1, but is based on the ε -greedy bandit algorithm (Bubeck & Cesa-Bianchi, 2012) instead of UCB1. That is, in every round, with probability $1 - \varepsilon$, the sampler with the smallest approximate block-KSD measure is selected, while with probability ε , a sampler is selected uniformly at random. In the experiments $\varepsilon = \frac{0.05}{\sqrt{t}}$ in round t .

We also compared the performance with that of NUTS. Figures 3.4 and 3.5 show the results for MH base samplers with $d = 2$ and as a function of d , respectively.

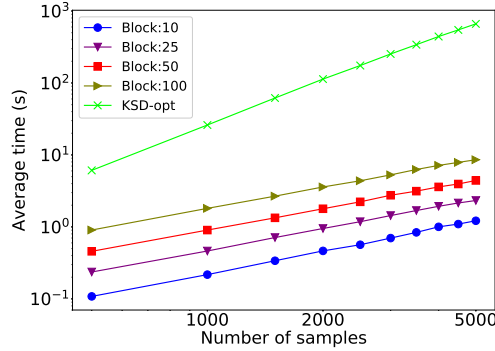


FIGURE 3.3: The average running time for different KSD based method.

One can observe that both bandit-based algorithms (i.e., KSD-UCB1 and KSD- ϵ -greedy) perform similarly, almost achieving the performance of the best base sampler. Interestingly, there is no significant difference between the two methods, and ϵ -greedy, which is an inferior bandit algorithm, seems to perform slightly better. This might indicate that in our case less exploration could be preferable. Also, it is interesting to note that the bandit method with the smaller batch size of 10 outperformed the one with bigger batch size of 100. Since we have already observed that the ordering of the samplers is not really affected by the batch size, this improved performance is most likely due to the increased number of decision points, which allows better adaptation. Note that given that the computational complexity of calculating the KSD measure is quadratic in the number of samples, smaller batch sizes have a huge advantage compared to bigger ones. In particular, in our experiment, the computational cost for batch size 10 is about two orders of magnitude smaller than for batch size 100. The running time of the different KSD computations are shown in Figure 3.3.

On the other hand, although our methods are competitive with the MH samplers, one can observe that NUTS significantly outperforms all of them, hence also their combinations. Furthermore, the reweighting mechanism of KSD-opt can provide a significant boost in performance for lower dimensions.

The performance of the aggregating sampling algorithm can be significantly improved by changing the base samplers. This is shown in Figures 3.6 and 3.7, where MH is replaced with the much better MALA samplers, making the best sampler and

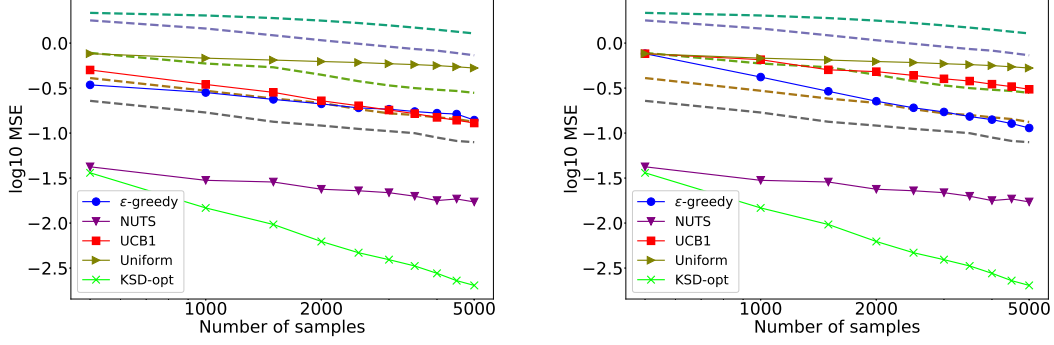


FIGURE 3.4: Unimodal case, Gaussian target distribution: MSE for different sample sizes with MH samplers in 2-dimensions ($d = 2$) with batch size 10 (left) and batch size 100 (right). The dashed lines without labels show the performance of the different MH samplers. The total number of samples is 5000 in each case.

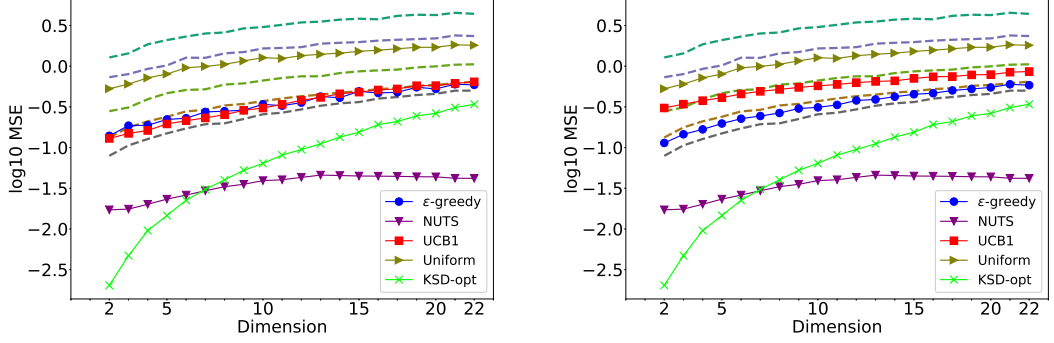


FIGURE 3.5: Unimodal case, Gaussian target distribution: MSE versus dimension d with MH samplers and batch size 10(left) and batch size 100 (right). The dashed lines without labels show the performance of the different MH samplers. The total number of samples is 5000 in each case.

our bandit-based sampling method competitive with NUTS. Interestingly, due to the better quality samples, in this case KSD-opt outperforms MALA for the whole range of dimensions (up to $d = 22$) we consider, although the performance difference vanishes as d increases, and indicates (as expected) that NUTS will be better for large values of d .

3.7.3 Separated Modes with One Sampler for Each Mode

In this section we consider the case of a multimodal target density (Gauss mixture) with separated modes, where we have one sampler for each mode. Here we run several

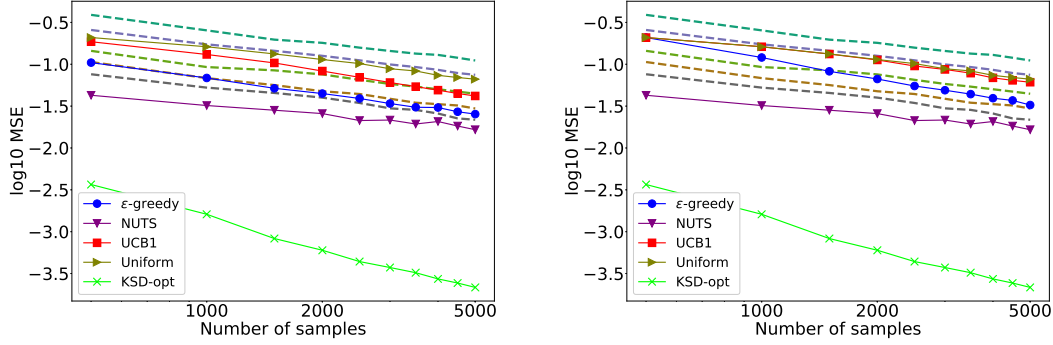


FIGURE 3.6: Unimodal case, Gaussian target distribution: MSE for different sample sizes with MALA samplers in 2-dimensions ($d = 2$) with batch size 10 (left) and batch size 100 (right). The dashed lines without labels show the performance of the different MALA samplers. The total number of samples is 5000 in each case.

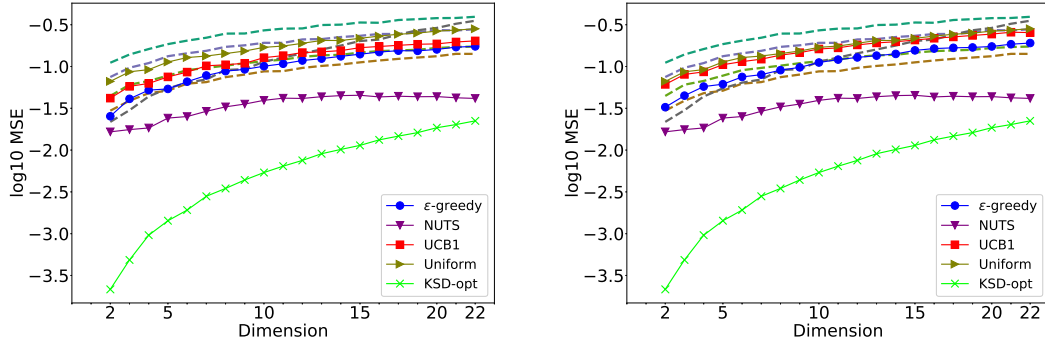


FIGURE 3.7: Unimodal case, Gaussian target distribution: MSE versus dimension d with MALA samplers and batch size 10 (left) and batch size 100 (right). The dashed lines without labels show the performance of the different MALA samplers. The total number of samples is 5000 in each case.

versions of KSD-UCB1-M, which differ in how the region (mode) I_t is selected in the algorithm. In particular, we consider the following methods for selecting I_t :

- Uniformly at random (this is called "Equal probability" in the figures).
 - Random, proportionally to the estimated weights w_i (or, equivalently, the estimated probability of the corresponding region), as described in Section 3.5.3 (w).
- In all the experiments, we used the Information Theoretical Estimators package of Szabó (2014) to calculate estimates of the Rényi entropy.

- I_t is selected as the sampler (region) with the largest average block-KSD measure (KSD).
- I_t is selected randomly with probability proportional to $w_i \tilde{S}_{i,n_i}$, as suggested by (3.21) (KSD. w).
- I_t based on the stratified sampling idea of (Carpentier et al., 2015), with probability proportional to $\hat{\sigma}_i w_i$, where w_i is the estimated weight and $\hat{\sigma}_i$ is the estimated standard deviation of the samples ($\sigma.w$)

For completeness, for the last reweighting step we conducted experiments with both the true weights and also with estimated weights.

We ran experiments for a target distribution with three separated modes, where the goal again was to estimate the mean of the distribution with respect to the mean squared error. We selected one sampler for each mode (with random parameters). The experiment were repeated 100 times, both for MH and MALA base samplers, with different random initializations for the MCMC chains. Finally, the whole process was repeated 20 times, drawing different parameters for the samplers. Figure 3.8 shows the results for one of the 20 settings with MH samplers (the results for all the 20 settings followed the same pattern); both with and without the a priori knowledge of the weights. The results with MALA instead of MH are presented in Figure 3.9. One can observe that, for known weights, the best method for selecting I_t in both experiments is the random choice with probability proportional to $w_i \tilde{S}_{i,n_i}$. On the other hand, when the weights are needed to be estimated, the performance of our more informed methods (i.e., all except for the uniform random selection) is approximately the same, which is due to the unavoidable errors in estimating the weights (this effect is more visible for small sample sizes).

3.7.4 Weight Estimation

A crucial step in the aggregation of the samples coming from different samplers is to estimate the weight of each mode. To this end, we repeated the same experiment with a uniformly random choice of I_t , but considering different methods for the final weight

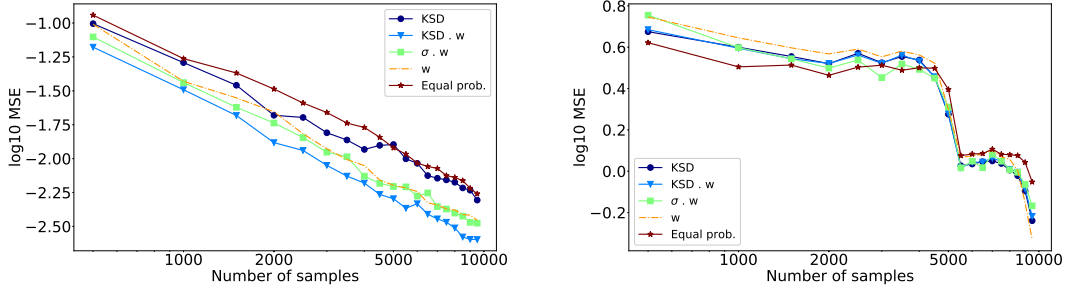


FIGURE 3.8: Multimodal distributions (Gauss mixtures): MSE for different sample sizes when the samplers are MH when the weight of each mode is known in advance (left); and when the weights are estimated with our proposed weight estimation method based on Rényi entropy with $\alpha = 0.95$ (right). The samples are taken from the following unnormalized distribution $p(x) = \sum_{i=1}^3 \beta_i \exp \{ (x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i) \}$ where $\beta = [0.5, 0.3, 0.2]$, $\Sigma_i = \sigma_i I$ with $\sigma = [0.9, 0.4, 0.5]$ and $\mu_1 = [6, 6]^\top$, $\mu_2 = [-6, 6]$ and $\mu_3 = [0, -6]$. On the right figure one can observe that the error slightly goes up after a sharp decline for all the methods (after about 5000 samples). This is due to a single chain that visited some areas of low probability that affected the weight estimation. The error starts to decline again when the chain returns to a high probability area.

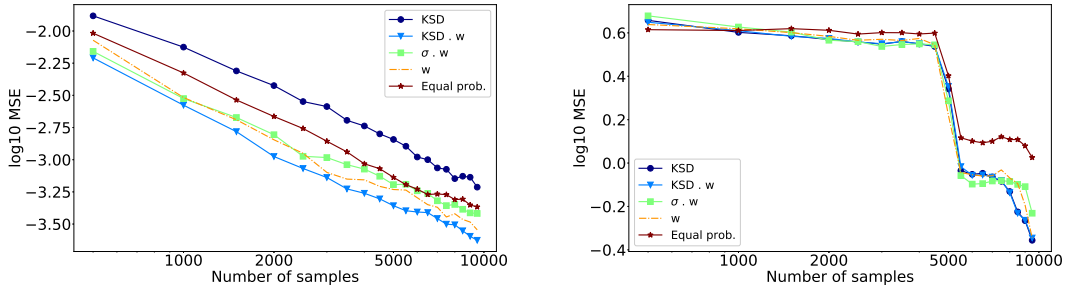


FIGURE 3.9: Multimodal distributions (Gauss mixtures): MSE for different sample sizes for the same target distribution as in Figure 3.8 with MALA base samplers when the weight of each mode is known in advance (left); and when the weights are estimated with our proposed weight estimation method based on Rényi entropy with $\alpha = 0.95$ (right).

estimation. In particular, we compare the Rényi-entropy-based estimates with different α parameters and the natural weight estimates for Gaussian mixtures based on estimating the covariance matrix for each mode (Scott et al., 2016). The results presented in Figure 3.10 show that the weight estimation is not really sensitive to the choice of α (the recommendation of Szabó, 2014 is to use values close to 1). Interestingly, in case of MALA base samplers, the Rényi-entropy-based methods outperform the ones designed

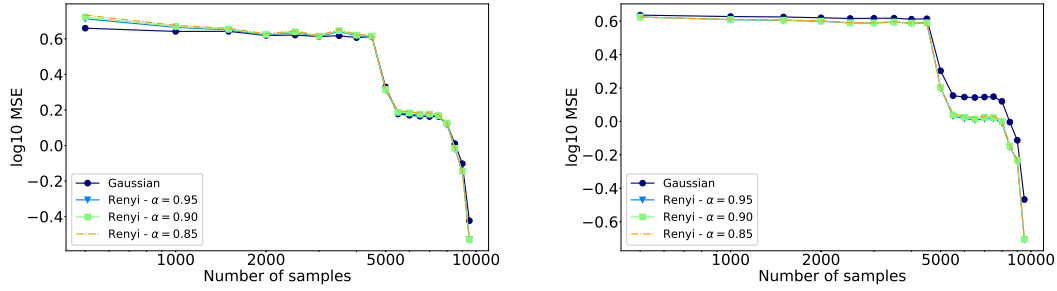


FIGURE 3.10: Multimodal distributions (Gauss mixtures): Mean squared error of different weight estimation methods for the setup in Figure 3.8 when the samplers receive equal budget. “Gaussian” refers to considering a Gaussian distribution for each mode and calculating its weight by estimating its covariance matrix, while “Rényi” refers to our proposed weight-estimation method based on Rényi entropy. Results for MH base samplers are shown on the left, while for MALA samplers on the right.

specifically for the Gaussian distribution, which is indeed the underlying distribution in our case. In a similar setup, Figure 3.11 compares several aggregation schemes, including the above tested Rényi- and, respectively, Gaussian-mixture-based combination, as well as uniform averaging and the computationally very expensive KSD-based reweighting scheme of Liu & Lee (2016). The results demonstrate that our method is able to outperform the others. On the other hand, extending the experiment to higher dimensions (and using NUTS as the base sampler, see Figure 3.12), the Gaussian and the Rényi-entropy-based estimators perform very similarly (recall that the Gaussian estimator is specifically fitted to the Gauss mixtures considered in the experiments).

3.7.5 The General Case: Unknown Number of Modes with Multiple Chains

In this section, we consider a realistic situation where we have access to an unnormalized density but the number of its separated regions/modes is unknown. Again, the goal is to estimate the mean of the distribution. Here we cannot guarantee that any MCMC chain used will only sample one mode; on the contrary, chains will usually move among different regions. We consider a Gaussian mixture model with 5 modes of random parameters, and run 10 base samplers whose parameters are also chosen randomly. We compare several combination methods discussed before: (i) each sampler

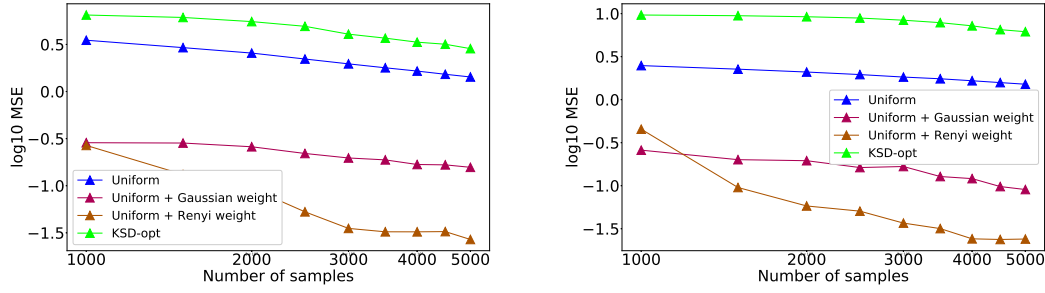


FIGURE 3.11: Unknown number of modes: Mean squared error of different weight estimation methods for a multimodal distribution in 2-dimensions ($d=2$) when combining 10 randomly initialized samplers. The samples are taken from a 2-dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode is selected uniformly at random from $[-5, 5]^2$ and the variance of each component is selected randomly from $[0.2, 1]$. “Gaussian” refers to considering a Gaussian distribution for each mode and calculating its weight by estimating its covariance matrix, while “Rényi” refers to our proposed weight-estimation method based on Rényi entropy with $\alpha = 0.99$. Results for MH base samplers are shown on the left, while for MALA samplers on the right.

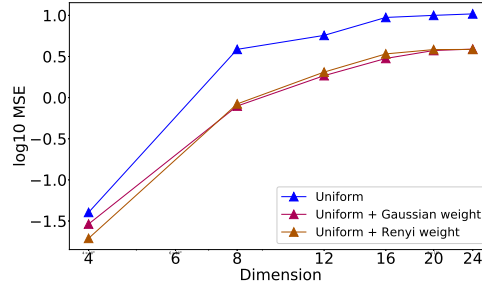


FIGURE 3.12: Unknown number of modes: Mean squared error of different weight estimation methods for a multimodal distribution in d -dimensions when combining 10 randomly initialized NUTS samplers. The samples are taken from a d -dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode is selected uniformly at random from $[-5, 5]^d$ and the variance of each component is selected randomly from $[0.5, 1]$. “Gaussian” refers to considering a Gaussian distribution for each mode and calculating its weight by estimating its covariance matrix, while “Rényi” refers to our proposed weight-estimation method based on Rényi entropy with $\alpha = 0.99$.

is used to generate the same number of samples (Uniform); (ii) the same as the previous, but the final clustering and reweighting step of KSD-MCMC-WR (Algorithm 10) is used in the end (Uniform+clustering); finally, four versions of KSD-MCMC-WR are considered where the bandit method is ε -greedy or UCB1, and the selection of I_t

is uniform at random (Equal probability) or random with probabilities proportional to $w_i \tilde{S}_{i,n_i}$. For clustering, we used the “Kmean” method of the scikit-learn package (Pedregosa et al., 2011).

After randomly choosing the distribution and the samplers, we considered them fixed and ran 100 experiments; in each of them the sampling methods are started from random initial points. The whole process was then repeated for 20 different sets of distributions and samplers. Figure 3.13 shows the results for two representative cases (out of 20) when MH is used as the base sampler. The top row corresponds to a situation when the modes are far from each other; in this case our proposed method significantly outperforms NUTS and also unweighted uniform usage of the samplers. The bottom row depicts a case when the modes are really close together, in which case NUTS performs the best. Interestingly, in these experiments the application of any bandit algorithm is not really beneficial as the best combination algorithm is essentially using the samplers uniformly and then reweighting the samples via our clustering method.

On the other hand, if the MH samplers are replaced with NUTS samplers, a more careful selection of which sampler to use becomes important. This is because these samplers explore individual modes efficiently, thus using multiple samplers for the same mode is really wasteful. Figure 3.14 shows the results for another representative example out of the 20 cases that we used in the previous experiment. The results indicate that our combination method (with various choices for the details) can significantly outperform NUTS. Also, as mentioned before, using bandit algorithms to select from the base samplers is much better than just choosing them uniformly.

Although the previous experiment demonstrated that the KSD-MCMC-WR is preferable to NUTS when the modes of the distribution are separated, it is not sufficient to argue that KSD-MCMC-WR is preferable to NUTS. In order to be able to make such a claim, we should also look at cases in favor of NUTS; the most favorable being the case of a unimodal distribution. To this end, we compared the performance of NUTS and KSD-MCMC-WR when sampling from a multivariate standard normal distribution. Figure 3.15 shows that running KSD-MCMC-WR is competitive with the individual NUTS instances in this case.

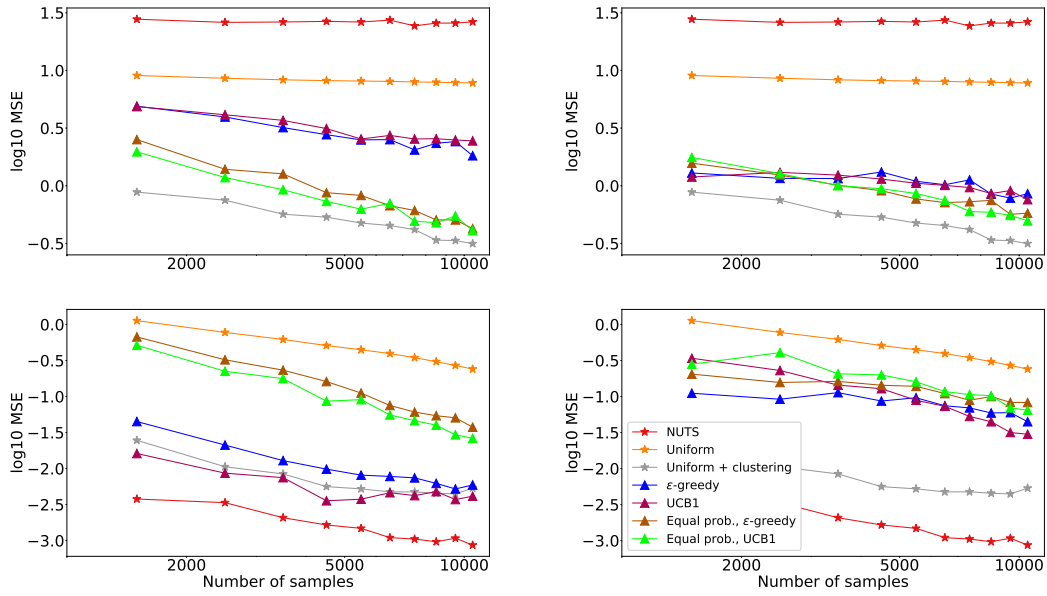


FIGURE 3.13: Unknown number of modes: MSE for different sample sizes when combining 10 randomly initialized MH samplers. The samples are taken from a 2-dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode is selected uniformly at random from $[-5, 5]^2$ and the variance of each component is selected randomly from $[0.2, 1]$. Two representative cases are shown: in the top row the modes are far from each other, in the bottom row the modes are close. The left column corresponds to batch size 10, while it is 100 for the right column. Rényi entropy with $\alpha = 0.99$ is used for weight estimation.

These results suggest that *one should always run KSD-MCMC-WR with NUTS base samplers*, as it provides significant improvements for multimodal distributions while remains a safe choice for the easier, unimodal cases.

3.7.6 Comparison with Parallel MCMC Methods

In the previous sections, we mostly compared our algorithms with MCMC methods (MH, MALA, and NUTS) that are only expected to work well for essentially unimodal densities (i.e., densities with a single “connected” region), and the vanilla parallelization (i.e., uniformly mixing multiple chains) is not sufficient to achieve good performance on multimodal problems. Therefore, we also compare our algorithms with two popular parallel MCMC methods, annealed sequential Monte Carlo with resampling (SMC) (Moral et al., 2006) and parallel tempering (PT) (Earl & Deem, 2005), which are proposed to deal with multimodal distributions. Following general practice

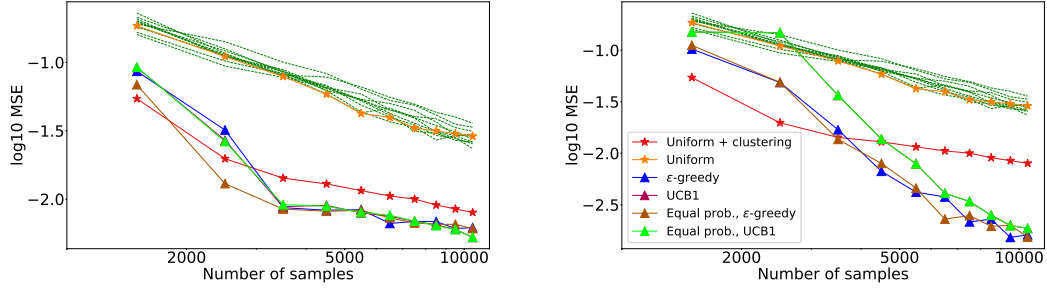


FIGURE 3.14: MSE for different sample sizes with separated modes: KSD-MCMC-WR with 10 randomly initialized NUTS samplers for an unknown number of modes. The batch size on the left figure is 10 and on the right figure is 100. The dashed green lines show the results for the individual NUTS samplers. Rényi entropy with $\alpha = 0.99$ is used for weight estimation. The problem setup is the same as the one in Figure 3.13.

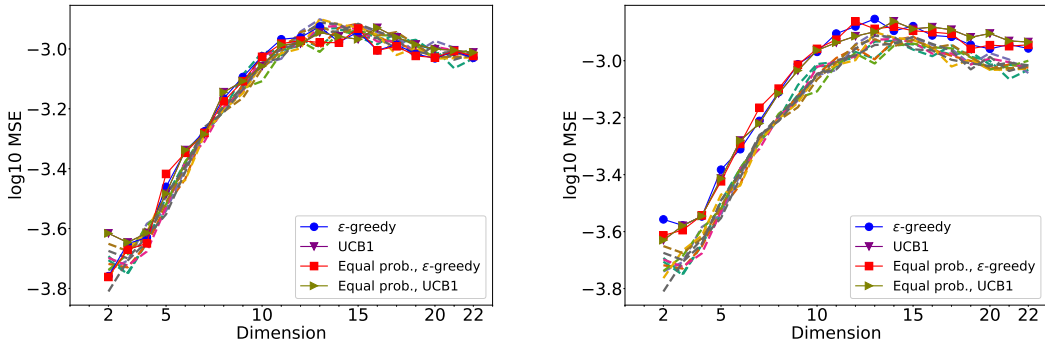


FIGURE 3.15: Running KSD-MCMC-WR for a multivariate standard normal distribution with 10 NUTS instances started with different (random) initializations. The batch size is 10 on the left and 100 on the right.

in the literature (similarly, e.g., to Liu & Lee, 2016) and in line with our previous experiments, we use the number of times the samplers calculate $p(x)$ for generating their final samples as a proxy for the runtime (a comparison based on actual runtimes is impossible since the algorithms are implemented using different packages). Throughout the experiments, for SMC and PT, we use 10 different temperature values for annealing with an exponential ladder of base $\sqrt{2}$ (with inverse temperature values $0, 2^{-4}, 2^{-3.5}, \dots, 1$). For SMC, we take one MCMC step between two resampling steps, and for PT 25 steps between two swapping attempts.

While our algorithm works best with NUTS as its base sampler, combining SMC and PT with NUTS is problematic: SMC uses a large initial population of samples at the

beginning and so the corresponding chains started from these points are too short (at least in the sample sizes we consider) for NUTS to be able to adapt. Similar problems occur with PT: since PT runs parallel chains whose states are swapped frequently, after each swap NUTS should be run for sufficiently long time to adapt to the new location, which slows down the process.¹⁴ Of course, one could try to avoid these problems by coming up with some clever method for integrating NUTS, but this is out of the scope of this work. Therefore, as a fair comparison, first we tested KSD-MCMC-WR, SMC and PT all with MALA as their base sampler in a 2-dimensional multimodal problem. The parameters of the target distribution were selected randomly (as described in Figure 3.16) and kept fixed during the experiment. Other target distributions yielded similar results. We ran SMC and PT with 10 different parameters of MALA, which were also used in KSD-MCMC-WR. The results, averaged over 100 runs with different initializations of the base samplers, are shown in Figure 3.16: it can be seen that our algorithms significantly outperformed PT and provided better results than SMC on average. The significance of our approach becomes clear by noticing that it is not even clear how one should set the MALA parameters for SMC in advance. It is interesting to note that in this experiment, replacing the bandit algorithms with a uniformly random selection of the chains followed by the Rényi entropy-based reweighting (“uniform+clustering” in the figure) has the best performance among all the algorithms with MALA base samplers, and in this case KSD-MCMC-WR works better with the smaller batch size 10 than with 100. This phenomenon is related to how easily the MALA samplers move from one mode to another, how it affects clustering, and how suitable the KSD measure becomes in this case to compare the sample quality of different chains clustered together, and is discussed in detail in the next section. For completeness, the results for KSD-MCMC-WR with NUTS as the base sampler are also shown in Figure 3.16, which confirms again that our methods work much better with NUTS, and our best method significantly outperformed all the other competitor algorithms.

In the second experiment, we compared the performance of our best algorithm, KSD-MCMC-WR with a NUTS base sampler, and that of SMC and PT with MALA.

¹⁴We have run PT with the pym3 implementation of NUTS, but the results were not good enough to be included.

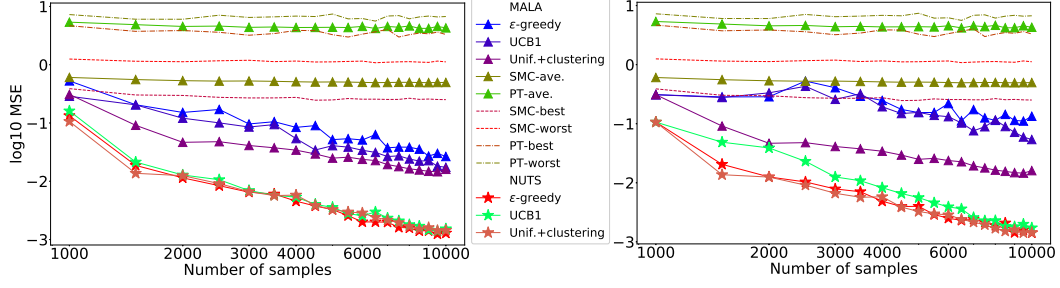


FIGURE 3.16: Unknown number of modes: MSE for different sample sizes when combining 10 randomly initialized MALA samplers for a 2-dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode was selected uniformly at random from $[-5, 5]^2$ and the variance of each component was selected uniformly from $[0.2, 1]$. SMC and PT were run separately for every parameter setting of MALA (10 different parameters) with the best, worst and average values reported, while KSD-MCMC-WR combined the same samplers. In addition, the results for KSD-MCMC-WR with 10 randomly initialized NUTS base samplers were included for completeness. Batch size for KSD-MCMC-WR is 10 on the left figure, 100 on the right. Each curve is obtained by averaging over 100 runs with different random initializations of the MALA/NUTS samplers (with the same parameters).

Figure 3.17 shows how the relative performance of the algorithms change with the dimension for randomly selected Gaussian mixture distributions (a single 24-dimensional distribution with 5 modes was selected randomly and its first d coordinates were used in the experiments for d -dimensions¹⁵): It can be seen that KSD-MCMC-WR with NUTS consistently outperformed SMC and PT, and the bandit selection methods (UCB1 and ϵ -greedy) in KSD-MCMC-WR performed better than the baseline uniform selection. Finally, the last experiment was repeated with different kernel widths, $h \in \{0.01, 0.1, 1, 10, 100\}$. The results in Figure 3.18 show that our proposed algorithm is insensitive to this parameter in a wide range, and also that the performance of the UCB1 and ϵ -greedy variants are very close to each other.¹⁶

3.7.7 Comparing the Components of KSD-MCMC-WR: Bandit Algorithms and Clustering-Reweighting

The two novel components of our final algorithm, KSD-MCMC-WR, are (i) using bandit algorithms with clustering and approximations to KSD and (ii) clustering and

¹⁵Other target distributions gave similar results.

¹⁶To reduce clutter, the results for SMC and PT are not shown in this figure.

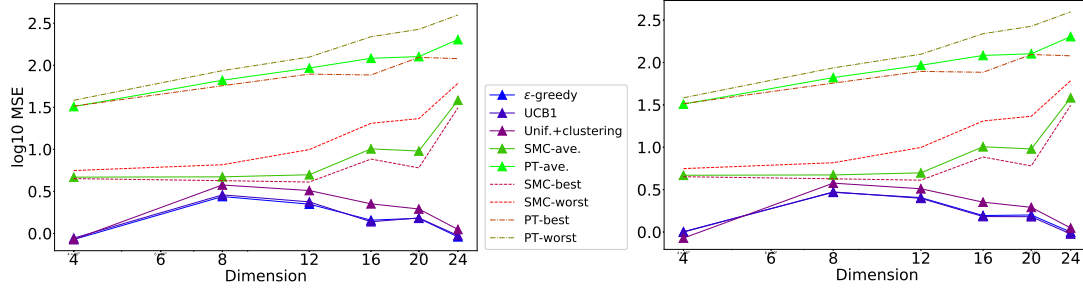


FIGURE 3.17: Unknown number of modes: MSE for different dimensions when using KSD-MCMC-WR with 10 randomly initialized NUTS samplers for a d -dimensional Gaussian mixture model with 5 isotropic modes. The mean of each mode was selected uniformly at random from $[-5, 5]^d$ and the variance of each component was selected randomly from $[0.5, 1]$. SMC and PT were run separately for every parameter setting of MALA (10 different parameters) with the best, worst and average values reported. Batch size for KSD-MCMC-WR is 10 on the left figure, 100 on the right. Each curve is obtained by averaging over 100 runs with different random initializations of the MALA/NUTS samplers (for the same parameters)

reweighting in the end via estimating Rényi-entropy. While we have already argued that (i) does not work without reweighting, to which our solution has been (ii), some of the experiments presented so far indicate that obtaining the same number of samples from each sampler followed by the reweighting process (ii) (the “uniform+clustering” method in our previous experiments) might lead to almost the same or even superior performance than that of KSD-MCMC-WR in certain cases (see Figure 3.13, Figure 3.14, the MALA-results in Figure 3.16, and Figure 3.17).

The reason behind this behavior seems to be a combination of how easily the samplers move from one mode to another, how this affects the clustering of the samplers, how meaningful it is to use the KSD to compare the sample quality of different samplers clustered together, and how many decisions the bandit algorithm can make. In essence, when some of the modes (“connected” regions) are close to each other so that samplers (particularly NUTS samplers) can easily go from one mode to another, samplers that explore these modes are often clustered together. At the same time, if the modes are far enough so that the KSD can signal good fit even if samples do not explore all these modes, the bandit algorithms can easily end up choosing samplers that only explore a small number of modes. Furthermore, even if the clusters are properly selected, when a sampler switches clusters, it may take a large number of steps until its KSD is comparable with that of other chains in the same group (since its KSD is

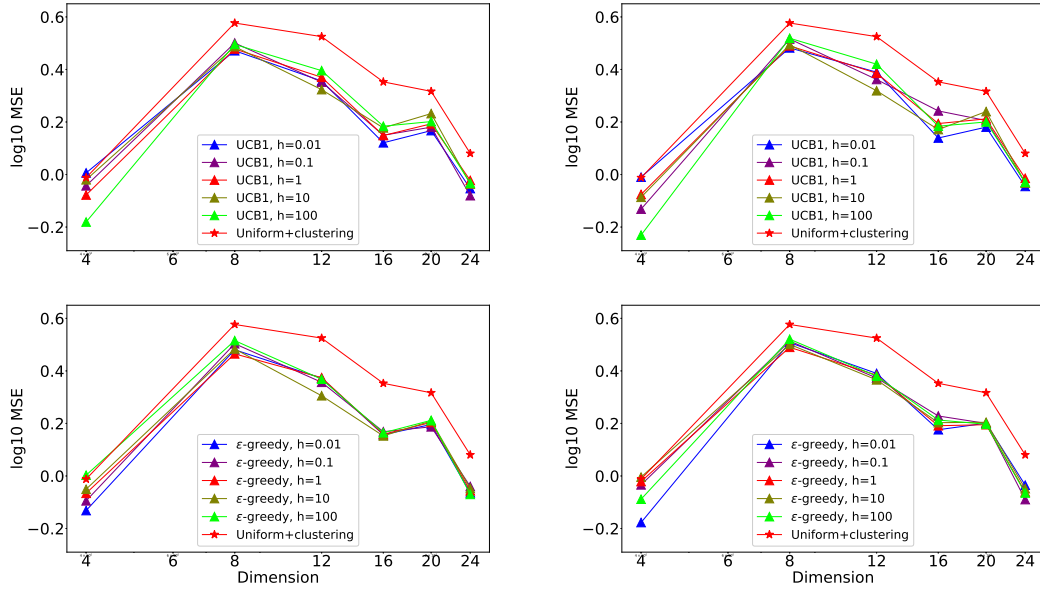


FIGURE 3.18: Changing the KSD kernel width h : MSE for different dimensions when using KSD-MCMC-WR with 10 randomly initialized NUTS samplers (the same setting as in Figure 3.17). Batch size for KSD-MCMC-WR is 10 on the left figures, 100 on the right; the performance of the UCB1 variant is shown in the top row, while that of the ε -greedy version in the bottom. Each curve is obtained by averaging over 100 runs with different initializations.

dominated by the sample quality in other modes). Using a smaller batch size in KSD-MCMC-WR can help create smaller clusters (and hence avoid the aforementioned problems). In addition, it also increases the number of decisions to be made by the bandit algorithm, which allows the bandit algorithm to adapt better (in cases when the KSD is a reasonable measure).

In the rest of this section we present some experiments verifying the above argument. Since we could only observe the superior behavior of the “uniform+clustering” algorithm for low-dimensional problems (this also holds for the relatively high, 18-dimensional experiments presented in the next section, where KSD-MCMC-WR consistently performs better), we consider a Gaussian mixture model with 20 modes of random parameters in 2D and run KSD-MCMC-WR with M NUTS samplers, $M \in \{10, 20, \dots, 100\}$.¹⁷ As before, after randomly choosing the distribution and the samplers, we considered them fixed and ran 100 experiments; in each of them the

¹⁷Note that here the number of samplers is much bigger (up to 100) than in any of our experiments.

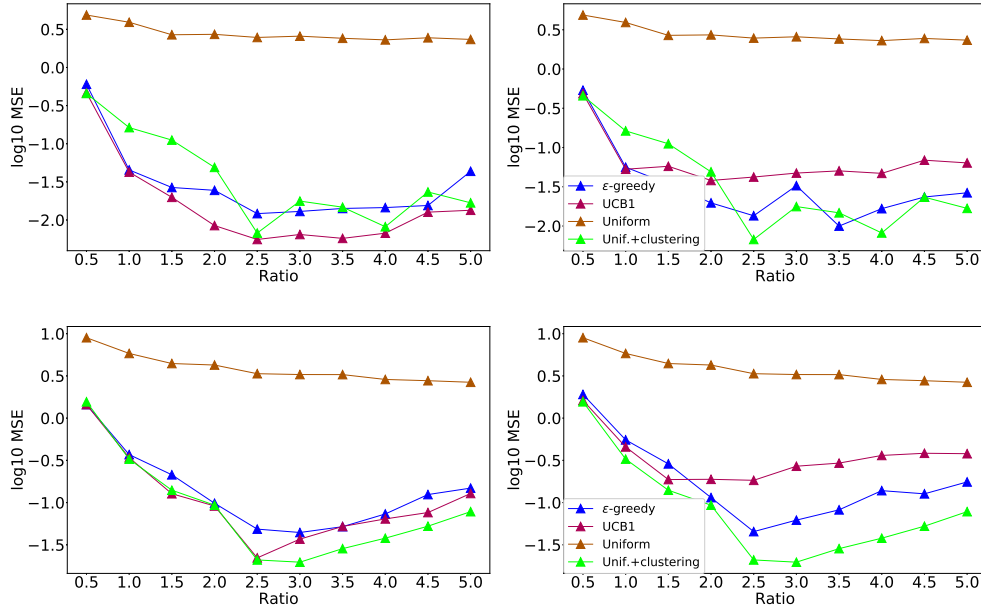


FIGURE 3.19: MSE for different number of NUTS samplers as a function of the number of samplers for a 2-dimensional Gaussian mixture model with 20 isotropic modes. The mean of each mode was selected uniformly at random from $[-10, 10]^2$ and the variance of each component was selected uniformly from $[0.2, 1]$. Two representative cases are shown: in the top row the modes are far from each other, in the bottom row some modes are close to each other. Batch size for KSD-MCMC-WR is 10 on the left figure and 100 on the right. Each curve is obtained by averaging over 100 runs with different random initializations of the NUTS samplers; the total number of samples is 20000 in each case.

sampling methods are started from random initial points. The whole process was then repeated for 10 different sets of distributions and samplers with the total number of samples being 20000 in each case. Figure 3.19 shows the results for two representative cases (out of 10). The top row corresponds to a case that the modes are far from each other and UCB1 significantly outperforms the uniform+clustering algorithm, except for few points. The bottom row depicts a case when the modes are close to each other, in which case UCB1 performs slightly worse than uniform+clustering. This is more pronounced for when the number of samplers is large, as in this case more modes are clustered together, while the samplers belonging to each cluster do not explore all the modes belonging to the cluster. Also, a large number of samplers often results in a larger number of clusters, increasing the decision space of the bandit algorithm, which deteriorates its performance in general.

3.7.8 Sensor Network Localization

In our last experiment we consider a simulated problem of *sensor network localization*. Following the setup of Ihler et al. (2005) (see also the work of Ahn et al., 2013), we consider N sensors which are independently placed in a planar region $[-L/2, L/2]^2$ for some $L > 0$ according to a density p_x . The distance between two sensors, t and u , are observed with probability $P_o(x_t, x_u) = \exp(-0.5\|x_t - x_u\|_2^2/R^2)$ (for some $R > 0$), and we define $o_{tu} = 1$ if the distance is observed and 0 otherwise. The observed distances are corrupted by independent zero-mean Gaussian noise n_{tu} with variance σ^2 : $d_{tu} = \|x_t - x_u\|_2 + n_{tu}$.

Given a set of observations $\{d_{tu}\}$ (which only includes t, u pairs with $o_{tu} = 1$), the joint distribution of the observations and locations is given by

$$p(\{x_t\}, \{o_{tu}\}, \{d_{tu}\}) = \prod_{t,u}^N \psi(x_t, x_u, o_{tu}) \prod_t^N p_x(x_t)$$

where, denoting the density of n_{tu} by p_{σ^2} ,

$$\psi(x_t, x_u, o_{tu}) = \begin{cases} P_o(x_t, x_u) p_{\sigma^2}(d_{tu} - \|x_t - x_u\|_2) & \text{if } o_{tu} = 1; \\ 1 - P_o(x_t, x_u) & \text{if } o_{tu} = 0. \end{cases}$$

The task is to infer the actual sensor locations from the observations. In the experiment we set $N = 8$, $R/L = 0.3$, $\sigma_v/L = 0.02$, $L = 10$, p_x is uniform, and we introduce three additional sensors with known locations, a.k.a. the anchor nodes, to avoid ambiguities due to translations, negations and rotations, as described by Ahn et al. (2013). Since several pairwise distances are not available, this setup results in a multimodal posterior distribution given the observations. The performance is measured by the Euclidean distance between the actual and the estimated locations from the sampling methods and we report the average results over 100 runs. We use KSD-MCMC-WR with NUTS and choosing each region uniformly at random. Figure 3.20 shows that our proposed algorithms significantly outperform all competitors, including single NUTS, parallel NUTS (labeled as uniform in the figures), SMC and PT for finding the sensor

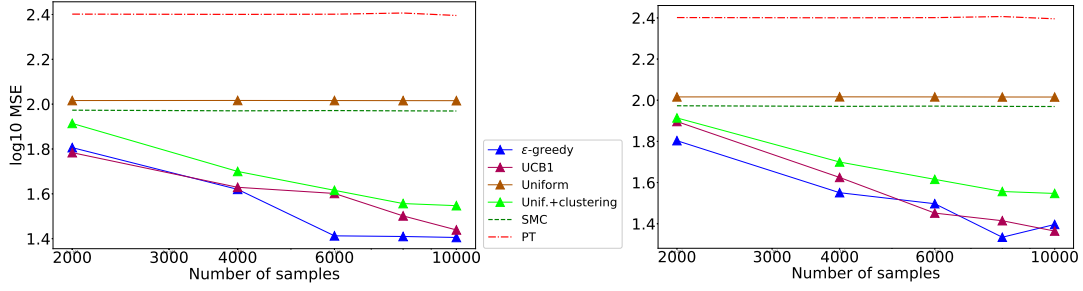


FIGURE 3.20: Running KSD-MCMC-WR for the sensor network localization problem with 10 NUTS instances started with different (random) initialization. The dashed green lines show the results for the individual NUTS samplers. Rényi entropy with $\alpha = 0.99$ is used for weight estimation. The batch size is 10 on the left and 100 on the right.

locations.¹⁸

3.8 Conclusions and Future Work

Selecting the best MCMC method and its best parameter setting for a given problem is a hard task. To solve these issues, we proposed an adaptive MCMC method that runs several chains in parallel, measures the quality of the samples from each sampler locally for every mode via kernel Stein discrepancy, and decides sequentially how to allocate the sampling budget among the different samplers using multi-armed bandit algorithms. The final step of our method is to combine the samples obtained by the different samplers: for this, we developed a novel weighting scheme based on Rényi-entropy estimation. Under “idealized” conditions, each part of the algorithm is asymptotically consistent and can be analyzed theoretically. Extensive experiments on several problems demonstrated that our proposed algorithm works well in both unimodal and multimodal problems, and it can perform particularly well when used in combination with NUTS as its base sampler.

Finally, in recent years, there has been a growing interest among researchers in deferent fields of science and engineering to applying machine learning techniques to their datasets. Generally, these datasets are very large and cannot be processed on

¹⁸Unfortunately, we could not reproduce the results of Ahn et al. (2013) because the paper does not provide enough details about how their algorithm is tuned.

a single machine. A common solution is to breakup the dataset to smaller datasets that can be managed by a single machine, run the algorithm of interest on the smaller datasets (i.e., this step can be on multiple machines), and combine the results of smaller datasets to form the final result. So, a natural next step for our algorithm as a machine learning technique is to extend it to be able to run parallel on multiple machines.

Chapter 4

Summary and Future Work

Statistical inference plays an important role in machine learning and probabilistic modeling. Doing exact inference is often not possible in practical scenarios. Therefore, approximate inference algorithms are of natural interest to deal with intractable inference tasks. Accordingly, the main research goals of this dissertation have been defined as (i) developing an efficient inference method for some special FHMMs; and (ii) automating parameter tuning and selection of MCMC methods to speed up their convergence.

FHMMs are widely used in energy disaggregation. E.g., the state-of-the-art NILM algorithm of Kolter & Jaakkola (2012) defines the energy disaggregation problem as an inference algorithm for FHMMs. However, the resulting model has a huge (factored) state space, making standard FHMM inference algorithms infeasible even for only a handful of appliances. To alleviate the problem, Kolter & Jaakkola (2012) solved the resulting non-convex problem approximately via QP. In this dissertation, we developed a scalable approximate inference algorithm for FHMMs based on a semidefinite relaxation combined with randomized rounding which significantly outperformed the state-of-the-art in our experiments. A crucial component of our solution is a scalable ADMM method that utilizes the special block-diagonal-like structure of the SDP relaxation and provides a good initialization for randomized rounding. We expect our method to be useful in solving other FHMM inference problems, as well as in large scale integer quadratic programming.

MCMC methods are widely used in high-dimensional Bayesian inference and maximum likelihood estimation problems, and computing analytically intractable integrals in econometrics, physics and computer science. Usually, these problems involve computing intractable expectation operations. MCMC methods compute these integrals by

simulating a Markov chain whose empirical distribution asymptotically converges to the target distribution defining the expectation. However, for practical simulation times, the empirical distribution of the samples generated by MCMC methods may be quite far from their stationary distribution, particularly, if the problem is multi-modal. Markov chains can easily get trapped in different modes for a long time, which prevents them from exploring the whole space. In addition, for a given problem, selecting the best MCMC method and its best parameter setting is a hard task since measuring the quality of the generated samples, which are correlated, is not easy. To solve these issues, in this dissertation, we proposed an adaptive MCMC method that runs several chains in parallel and decides sequentially how to allocate the sampling budget among the different samplers, using multi-armed bandit algorithms. We expect each sampler to explore some part of the state space, not necessarily the whole. The proposed algorithm uses kernel Stein discrepancy to measure the quality of samples from each sampler locally, and assigns resources to each sampler accordingly. Finally, the samples generated by the different samplers must be combined in an unbiased fashion. To do so, we developed a novel weighting scheme based on Rényi-entropy estimation, which might be of independent interest. Through several experiments, we showed that our proposed algorithm provides state-of-the-art performance when used in combination with NUTS as its base sampler, outperforming competitor algorithms for multimodal problems, while paying only a slight overhead in case of unimodal target distributions.

Developing approximate inference algorithms for graphical models requires a considerable amount of time and efforts, and the best algorithms to any given problem are usually problem-specific. We had the same experience in developing an approximate inference solution for FHMMs in Chapter 2, where our final solution heavily exploited the problem structure. An alternative approach to our hand-designed algorithm could be sampling. While sampling algorithms are guaranteed to converge to the true solution in the limit of infinite data, their behavior can be significantly different for practical sample sizes. Therefore, these algorithms are typically hard to use in a black-box manner, and usually some domain-specific knowledge is used in the design of the final algorithms and in tuning their parameters properly, as discussed in Chapter 3. In

fact, the main motivation behind developing the adaptive sampling algorithm in Chapter 3 was to develop a sampling algorithm that can adapt automatically to the problem at hand, easing the burden of a practitioner to hand-design and hand-tune a method for every problem instance. The proposed adaptive method solves this issue to a great extent, achieving or improving the state-of-the-art in several scenarios without hand-tuning the algorithm's parameters. At the same time, it is not expected to work well out of the box for graphical models in general, since sampling all variables at the same time requires sampling from a very high dimensional space. In Chapter 2 we applied multiple tricks to resolve this problem and estimate the variables sequentially; how to incorporate similar techniques into our sampling method, and hence how to extend its applicability to graphical models in general is one of the most important question for our future research. Another interesting research direction is to provide a careful theoretical analysis of our sampling method and proving its consistency, complementing our analysis where each part was separately considered under idealized conditions. Finally, we believe that our approach can be applied in the future to adaptively tune the parameters of other classes of sampling and inference algorithms.

Bibliography

- Ahn, Sungjin, Chen, Yutian, and Welling, Max. Distributed and adaptive darting monte carlo through regenerations. volume 31 of *Proceedings of Machine Learning Research*, pp. 108–116, 2013.
- Anandkumar, Animashree, Hsu, Daniel, and Kakade, Sham M. A Method of Moments for Mixture Models and Hidden Markov Models. In *COLT*, volume 23, pp. 33.1–33.34, 2012.
- Andrieu, Christophe and Thoms, Johannes. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, December 2008. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-008-9110-y.
- Andrieu, Christophe, De Freitas, Nando, Doucet, Arnaud, and Jordan, Michael I. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.
- Angelino, E., Johnson, M. J., and Adams, R. P. *Patterns of Scalable Bayesian Inference*. Now Publishers Inc., Hanover, MA, USA, 2016.
- Atchadé, Yves F., Roberts, Gareth O., and Rosenthal, Jeffrey S. Towards optimal scaling of metropolis-coupled Markov chain Monte Carlo. *Statistics and Computing*, 21(4):555–568, October 2011. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-010-9192-1.
- Auer, Peter, Cesa-Bianchi, Nicolo, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.
- Baele, Guy, Lemey, Philippe, Rambaut, Andrew, and Suchard, Marc A. Adaptive mcmc in bayesian phylogenetics: an application to analyzing partitioned data in beast. *Bioinformatics*, 2 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx088.
- Bakir, Gökhan H., Hofmann, Thomas, Schölkopf, Bernhard, Smola, Alexander J., Taskar, Ben, and Vishwanathan, S. V. N. *Predicting Structured Data (Neural Information Processing)*. MIT Press, 2007.

- Barber, David. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.
- Bishop, Christopher M. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.
- Boucheron, Stéphane, Lugosi, Gábor, and Massart, Pascal. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
- Boyd, Stephen, Parikh, Neal, Chu, Eric, Peleato, Borja, and Eckstein, Jonathan. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *FTML*, 3(1):1–122, 2011.
- Brooks, Steve, Gelman, Andrew, Jones, Galin L., and Meng, Xiao-Li. *Handbook of markov chain monte carlo*. Chapman and Hall/CRC, 2011.
- Bubeck, Sébastien and Cesa-Bianchi, Nicolò. Regret Analysis of Stochastic and Non-stochastic Multi-armed Bandit Problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012. ISSN 1935-8237, 1935-8245. doi: 10.1561/22000000024.
- Bédard, Mylène. Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234. *Stochastic Processes and their Applications*, 118(12):2198–2222, December 2008. ISSN 03044149. doi: 10.1016/j.spa.2007.12.005.
- Carpentier, Alexandra, Munos, Remi, and Antos, András. Adaptive strategy for stratified Monte Carlo sampling. *Journal of Machine Learning Research*, 16:2231–2271, 2015.
- Cauchemez, S., Carrat, F., Viboud, C., Valleron, A. J., and Boëlle, P. Y. A bayesian mcmc approach to study transmission of influenza: application to household longitudinal data. *Statistics in Medicine*, 23(22):3469–3487, 2004. ISSN 1097-0258. doi: 10.1002/sim.1912.
- Chahine, Khaled, Drissi, Khalil El Khamlichi, Pasquier, Christophe, Kerroum, Kamal, Faure, Claire, Jouannet, Thierry, and Michou, Michel. Electric Load Disaggregation in Smart Metering Using a Novel Feature Extraction Method and Supervised Classification. *Energy Procedia*, 6:627–632, 2011. ISSN 18766102. doi: 10.

- 1016/j.egypro.2011.05.072. URL <http://linkinghub.elsevier.com/retrieve/pii/S1876610211014834>.
- Chang, Hsueh-Hsien, Chen, Kun-Long, Tsai, Yuan-Pin, and Lee, Wei-Jen. A New Measurement Method for Power Signatures of Nonintrusive Demand Monitoring and Load Identification. *IEEE T. on Industry Applications*, 48:764–771, 2012.
- Chang, Joseph T. Full reconstruction of markov models on evolutionary trees: Identifiability and consistency. *Mathematical biosciences*, 137:51–73, 11 1996. doi: 10.1016/S0025-5564(96)00075-2.
- Chwialkowski, Kacper, Strathmann, Heiko, and Gretton, Arthur. A Kernel Test of Goodness of Fit. *arXiv:1602.02964 [stat]*, February 2016. arXiv: 1602.02964.
- Cybenko, George and Crespi, Valentino. Learning Hidden Markov Models Using Nonnegative Matrix Factorization. *IEEE Transactions on Information Theory*, 57(6):3963–3970, June 2011. ISSN 0018-9448, 1557-9654. doi: 10.1109/TIT.2011.2132490. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5773017>.
- Dong, Ming, Meira, Paulo C. M., Xu, Wilsun, and Freitas, Walmir. An Event Window Based Load Monitoring Technique for Smart Meters. *IEEE Transactions on Smart Grid*, 3(2):787–796, June 2012.
- Dongarra, J. and Sullivan, F. Guest editors introduction to the top 10 algorithms. *Computing in Science Engineering*, 2(1):22–23, Jan 2000. ISSN 1521-9615. doi: 10.1109/MCISE.2000.814652.
- Doucet, Arnaud, De Freitas, Nando, and Gordon, Neil. An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001a.
- Doucet, Arnaud, Defreitas, Nando, and Gordon, Neil. *An Introduction to Sequential Monte Carlo Methods*. Springer-Verlag,, New York, 2001b.
- Earl, David J. and Deem, Michael W. Parallel tempering: theory, applications, and new perspectives. *Physical Chemistry Chemical Physics (PCCP)*, 7 23:3910–6, 2005.

- Egarter, Dominik, Bhuvana, Venkata Pathuri, and Elmenreich, Wilfried. PALDi: On-line Load Disaggregation via Particle Filtering. *IEEE Transactions on Instrumentation and Measurement*, 64(2):467–477, 2015.
- Figueiredo, Marisa, de Almeida, Ana, and Ribeiro, Bernardete. Home Electrical Signal Disaggregation for Non-intrusive Load Monitoring (NILM) Systems. *Neurocomputing*, 96:66–73, November 2012.
- Ghahramani, Z. and Jordan, M.I. Factorial Hidden Markov Models. *Machine learning*, 29(2):245–273, 1997.
- Ghahramani, Zoubin. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14541.
- Girolami, Mark and Calderhead, Ben. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011. doi: 10.1111/j.1467-9868.2010.00765.x.
- Goemans, Michel X. and Williamson, David P. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. of the ACM*, 42(6):1115–1145, 1995.
- Gorham, Jack, Duncan, Andrew B., Vollmer, Sebastian J., and Mackey, Lester. Measuring sample quality with diffusions. *arXiv preprint arXiv:1611.06972*, 2016.
- Gorham, Jackson and Mackey, Lester. Measuring sample quality with Stein’s method. In *Advances in Neural Information Processing Systems*, pp. 226–234, 2015.
- Gorham, Jackson and Mackey, Lester. Measuring sample quality with kernels. In *ICML*, pp. 1292–1301, 2017.
- Guo, Zhenyu, Wang, Z. Jane, and Kashani, Ali. Home Appliance Load Modeling From Aggregated Smart Meter Data. *IEEE Transactions on Power Systems*, 30(1): 254–262, January 2015.
- Haario, Heikki, Saksman, Eero, and Tamminen, Johanna. An Adaptive Metropolis Algorithm. *Bernoulli*, 7(2):223, April 2001. ISSN 13507265. doi: 10.2307/3318737.

- Haario, Heikki, Laine, Marko, Mira, Antonietta, and Saksman, Eero. DRAM: Efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, December 2006. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-006-9438-0.
- Hero, A. O. and Michel, O. J. J. Asymptotic theory of greedy approximations to minimal k-point random graphs. *IEEE Transactions on Information Theory*, 45(6): 1921–1938, 1999.
- Hoffman, Matthew D. and Gelman, Andrew. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.
- Hsu, Daniel, Kakade, Sham M., and Zhang, Tong. A spectral algorithm for learning Hidden Markov Models. *Journal of Computer and System Sciences*, 78(5):1460–1480, September 2012. ISSN 00220000. doi: 10.1016/j.jcss.2011.12.025. URL <http://linkinghub.elsevier.com/retrieve/pii/S0022000012000244>.
- Ihler, A. T., Fisher, J. W., Moses, R. L., and Willsky, A. S. Nonparametric belief propagation for self-localization of sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):809–819, 2005.
- Johannes, Michael and Polson, Nicholas. {CHAPTER} 13 - {MCMC} methods for continuous-time financial econometrics. In AÏT-SAHALIA, YACINE and HANSEN, LARS PETER (eds.), *Handbook of Financial Econometrics: Applications*, volume 2 of *Handbooks in Finance*, pp. 1 – 72. Elsevier, San Diego, 2010. doi: <https://doi.org/10.1016/B978-0-444-53548-1.50003-9>.
- Jordan, Michael I. Graphical models. *Statist. Sci.*, 19(1):140–155, 02 2004. doi: 10.1214/088342304000000026. URL <https://doi.org/10.1214/088342304000000026>.
- Kelly, Jack and Knottenbelt, William. Neural NILM: Deep Neural Networks Applied to Energy Disaggregation. In *BuildSys*, pp. 55–64, 2015a.
- Kelly, Jack and Knottenbelt, William. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Scientific*

- Data*, 2:150007, March 2015b. ISSN 2052-4463. doi: 10.1038/sdata.2015.7. URL <http://www.nature.com/articles/sdata20157>.
- Kim, Hyungsul, Marwah, Manish, Arlitt, Martin F., Lyon, Geoff, and Han, Jiawei. Unsupervised Disaggregation of Low Frequency Power Measurements. In *ICDM*, volume 11, pp. 747–758, 2011.
- Kodali, A., Pattipati, K., and Singh, S. Coupled factorial hidden markov models (cfhmm) for diagnosing multiple and coupled faults. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(3):522–534, May 2013.
- Kokkala, Juho and Särkkä, Simo. Combining particle MCMC with rao-blackwellized monte carlo data association for parameter estimation in multiple target tracking. *Digital Signal Processing*, 47:84 – 95, 2015. Special Issue in Honour of William J. (Bill) Fitzgerald.
- Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, 2009.
- Kolter, J. Zico and Jaakkola, Tommi. Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation. In *AISTATS*, pp. 1472–1482, 2012.
- Kolter, J. Zico and Johnson, Matthew J. REDD: A Public Data Set for Energy Disaggregation Research. In *Workshop on Data Mining Applications in Sustainability (SIGKDD)*, pp. 59–62, 2011.
- Kolter, J. Zico, Batra, Siddharth, and Ng, Andrew Y. Energy Disaggregation via Discriminative Sparse Coding. In *Advances in Neural Information Processing Systems*, pp. 1153–1161, 2010.
- Kontorovich, A., Nadler, B., and Weiss, R. On Learning Parametric-Output HMMs. In *ICML*, pp. 702–710, 2013.
- Lewis, Antony and Bridle, Sarah. Cosmological parameters from cmb and other data: A monte carlo approach. *Phys. Rev. D*, 66:103511, Nov 2002. doi: 10.1103/PhysRevD.66.103511.

- Liang, Jian, Ng, Simon K. K., Kendall, Gail, and Cheng, John W. M. Load Signature Study-Part II: Disaggregation Framework, Simulation, and Applications. *IEEE Transactions on Power Delivery*, 25(2):561–569, April 2010.
- Lin, Yu-Hsiu and Tsai, Men-Shen. Application of Neuro-Fuzzy Pattern Recognition for Non-intrusive Appliance Load Monitoring in Electricity Energy Conservation. In *FUZZ-IEEE*, pp. 1–7, 2012.
- Liu, Qiang and Lee, Jason D. Black-box importance sampling. *arXiv preprint arXiv:1610.05247*, 2016. URL <https://arxiv.org/abs/1610.05247>.
- Liu, Qiang, Lee, Jason D., and Jordan, Michael I. A kernelized Stein discrepancy for goodness-of-fit tests. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.
- Mahendran, Nimalan, Wang, Ziyu, Hamze, Firas, and De Freitas, Nando. Adaptive MCMC with Bayesian Optimization. In *AISTATS*, volume 22, pp. 751–760, 2012.
- Malick, Jérôme, Povh, Janez, Rendl, Franz, and Wiegeler, Angelika. Regularization Methods for Semidefinite Programming. *SIAM Journal on Optimization*, 20(1):336–356, January 2009. ISSN 1052-6234, 1095-7189.
- Manton, Jonathan H. and Amblard, Pierre-Olivier. A primer on reproducing kernel hilbert spaces. *Foundations and Trends® in Signal Processing*, 8(1–2):1–126, 2015. ISSN 1932-8346. doi: 10.1561/20000000050.
- Mattfeld, Carl. Implementing spectral methods for hidden Markov models with real-valued emissions. *arXiv preprint arXiv:1404.7472*, 2014.
- Mbalawata, Isambi S., Särkkä, Simo, Vihola, Matti, and Haario, Heikki. Adaptive Metropolis algorithm using variational Bayesian adaptive Kalman filter. *Computational Statistics & Data Analysis*, 83:101–115, March 2015. ISSN 01679473. doi: 10.1016/j.csda.2014.10.006.
- Moral, Pierre Del, Doucet, Arnaud, and Jasra, Ajay. Sequential monte carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 68(3): 411–436, 2006.

- Mykland, Per, Tierney, Luke, and Yu, Bin. Regeneration in markov chain samplers. *Journal of the American Statistical Association*, 90(429):233–241, 1995. ISSN 01621459.
- Neal, Radford M. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 06 2012. doi: 10.1201/b10905-6.
- Neiswanger, W., Wang, C., and Xing, E. P. Asymptotically exact, embarrassingly parallel mcmc. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI, pp. 623–632, Arlington, Virginia, United States, 2014. AUAI Press. ISBN 978-0-9749039-1-0.
- Nemeth, Christopher and Sherlock, Christopher Gerrard. Merging mcmc subposteriors through gaussian-process approximations. *Bayesian Analysis*, 13(2):507–530, 3 2018. ISSN 1936-0975.
- Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, 2004.
- Neufeld, James. *Adaptive Monte Carlo Integration*. PhD thesis, University of Alberta, 2016.
- Neufeld, James, György, András, Schuurmans, Dale, and Szepesvári, Csaba. Adaptive Monte Carlo via bandit allocation. In *ICML*, pp. 1944–1952, 2014.
- Nowozin, Sebastian and Lampert, Christoph H. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6 (3–4):185–365, 2011. ISSN 1572-2740. doi: 10.1561/06000000033.
- Oliver, N. M., Rosario, B., and Pentland, A. P. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, Aug 2000. ISSN 0162-8828. doi: 10.1109/34.868684.
- Owen, Art B. *Monte Carlo theory, methods and examples*. 2013. URL <http://statweb.stanford.edu/~owen/mc/>.

- Pál, Dávid, Póczos, Barnabás, and Szepesvári, Csaba. Estimation of rényi entropy and mutual information based on generalized nearest-neighbor graphs. *NIPS*, pp. 1849–1857, 2010.
- Parikh, Neal and Boyd, Stephen. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231, 2013. URL <http://www.ebookcs.com/download/zdm9Z3/proximal-algorithms-stanford-university.pdf>.
- Park, Jaehyun and Boyd, Stephen. A Semidefinite Programming Method for Integer Convex Quadratic Minimization. *arXiv preprint arXiv:1504.07672*, 2015.
- Parson, Oliver, Ghosh, Siddhartha, Weal, Mark, and Rogers, Alex. Non-intrusive load monitoring using prior models of general appliance types. 2012. URL <http://www.orchid.ac.uk/eprints/49/>.
- Pasarica, Cristian and Gelman, Andrew. Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, pp. 343–364, 2010.
- Pearson, Karl. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894. URL <http://www.jstor.org/stable/90667>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Rennie, Steven J., Hershey, John R., and Olsen, Peder. Single-channel speech separation and recognition using loopy belief propagation. In *ICASSP*, pp. 3845–3848, 2009a.
- Rennie, Steven J., Hershey, John R., and Olsen, Peder A. Variational loopy belief propagation for multi-talker speech recognition. In *INTERSPEECH*, pp. 1331–1334, 2009b.
- Robert, Christian P. and Casella, George. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387212396.

- Roberts, G. O. and Rosenthal, J. S. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 2002.
- Roberts, Gareth O. and Rosenthal, Jeffrey S. Optimal Scaling for Various Metropolis-Hastings Algorithms. *Statistical Science*, 16(4):351–367, 2001. ISSN 0883-4237.
- Roberts, Gareth O., Gelman, Andrew, Gilks, Walter R., and others. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.
- Salvatier, J., V Wiecki, T., and Fonnesbeck, C. Probabilistic programming in python using pymc3. 2016. URL <https://doi.org/10.7717/peerj-cs.55>.
- Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. Bayes and big data: The consensus monte carlo algorithm. *International Journal of Management Science and Engineering Management*, 11:78–88, 2016.
- Sejdinovic, Dino, Strathmann, Heiko, Garcia, Maria Lomeli, Andrieu, Christophe, and Gretton, Arthur. Kernel Adaptive Metropolis-Hastings. In *ICML*, pp. 1665–1673, 2014.
- Shimony, Solomon Eyal. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68(2):399 – 410, 1994.
- Siddiqi, Sajid M., Boots, Byron, and Gordon, Geoffrey J. Reduced-Rank Hidden Markov Models. In *AISTATS*, pp. 741–748, 2010. URL <http://www.jmlr.org/proceedings/papers/v9/siddiqi10a/siddiqi10a.pdf>.
- Sokal, Alan D. *Monte Carlo methods in statistical mechanics: foundations and new algorithms*. Troisieme cycle de la physique en Suisse Romande, 1989.
- Szabó, Zoltán. Information theoretical estimators toolbox. *Journal of Machine Learning Research*, 15:283–287, 2014.
- Wainwright, Martin J. and Jordan, Michael I. Graphical Models, Exponential Families, and Variational Inference. *FTML*, 1(1–2):1–305, 2007.

- Wang, Ziyu, Mohamed, Shakir, and De Freitas, Nando. Adaptive Hamiltonian and Riemann manifold Monte Carlo samplers. In *International Conference on Machine Learning (ICML)*, pp. 1462–1470, 2013.
- Weiss, Markus, Helfenstein, Adrian, Mattern, Friedemann, and Staaake, Thorsten. Leveraging smart meter data to recognize home appliances. In *PerCom*, pp. 190–197, 2012.
- Wen, Zaiwen, Goldfarb, Donald, and Yin, Wotao. Alternating direction augmented Lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, December 2010.
- Yang, Jinyoung and Rosenthal, Jeffrey S. Automatically tuned general-purpose mcmc via new adaptive diagnostics. *Computational Statistics*, 32(1):315–348, March 2017.
- Yu, Liu, Li, Haibin, Feng, Xiaowei, and Duan, Jizhong. Nonintrusive appliance load monitoring for smart homes: recent advances and future issues. *IEEE Instrumentation & Measurement Magazine*, 19(3):56–62, 2016. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7477956.
- Zhong, Mingjun, Goddard, Nigel, and Sutton, Charles. Signal Aggregate Constraints in Additive Factorial HMMs, with Application to Energy Disaggregation. In *NIPS*, pp. 3590–3598, 2014.
- Zia, Tehseen, Bruckner, Dietmar, and Zaidi, Adeel. A hidden Markov model based procedure for identifying household electric loads. In *IECON*, pp. 3218–3223, 2011.
- Łatuszyński, Krzysztof, Roberts, Gareth O., and Rosenthal, Jeffrey S. Adaptive Gibbs samplers and related MCMC methods. *The Annals of Applied Probability*, 23(1): 66–98, February 2013. ISSN 1050-5164. doi: 10.1214/11-AAP806.