Modelling, Inference and Optimization in Probabilistic Machine Learning

Xiaoyu Lu

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Xiaoyu Lu
January 2019

Acknowledgement

I would like to thank many people who have helped me through my PhD study. In particular, I would like to thank my supervisor Professor Yee Whye Teh for being such an inspirational person and guiding me through the PhD. I would also like to thank my collaborators and colleagues: Tom Rainforth, Hyunjik Kim, Valerio Perrone, Leonard Hasenclever, Seth Flaxman, Sebastian Vollmer; and industrial supervisors during my internships at Amazon and Microsoft Research: Javier Gonzalez Hernandez, Zhenwen Dai, Neil Lawrence, Katja Hoffman, Jan Stuehmer and Sam Devlin and many other researchers for their helpful comments and discussions. Thank Yingzhen Li for her insightful advice on the structure of this thesis.

I would also like to thank my parents, uncles, aunties, cousins and many other relatives for their emotional and financial support during my journey of studying abroad. I thank my dear friends for the good company and helpful discussion which means a lot to me. Furthermore, I would like to thank all of my dance teachers and partners who have made dancing an integrated part of my life.

Lastly, I would like to thank Oxford Clarendon and PAG Oxford fund for sponsoring my 4 years of PhD study, making the whole experience invaluable and an everlasting memory.

Abstract

Bayesian machine learning has gained tremendous attention in the machine learning community over the past few years. Bayesian methods offer a coherent reasoning for quantifying uncertainties in the decision making procedure, based on the Bayes rule. One of the core advantages of Bayesian methods is the separation of modelling and inference. In other words, the likelihood models are completely independent of the computation of the posterior distribution of the parameters.

There are many Bayesian models that are widely used in the machine learning community. For example, non-parametric models such as Gaussian Processes and Dirichlet Processes are flexible models which are able to capture and learn the structure of the data. Bayesian deep learning models, which are based on neural networks, are another example of flexible Bayesian models that are rich enough to represent non-linear structures in the data.

The process of inferring the posterior lies at the center of Bayesian inference. When computing the posterior distribution exactly is not feasible, due to intractability of the posterior and the computational or memory constraints, approximate Bayesian inference comes to play.

In this PhD thesis, I develop and investigate various Bayesian modelling and inference techniques and apply them to multiple interesting domains and tasks. We begin with *Tucker Gaussian Processes(TGP)*, a class of flexible non-parametric models based on *Gaussian Processes (GP)*. We apply the method to 1) regression problems on structured input data, and 2) collaborative filtering problems where TGP offers an elegant way of incorporating side information. We demonstrate superior results compared with benchmarks on a number of examples across different domains.

A closely related line of research based on GPs is *Bayesian Optimization (BO)*. It is a black-box optimizer where one optimizes an objective function through subsequent queries about next input locations to be evaluated

at. However, this method does not work well when the input space is non-Euclidean or combinatorial. We alleviate the problem by learning a low dimensional Euclidean representation of the combinatorial input space with variational inference, using *Variational Auto-encoder (VAE)*. The optimization can then be conducted on the low dimensional embedding instead. We apply our method to *Automatic Statistician* and natural scene understanding, which give promising results.

For approximate Bayesian inference, we first propose an algorithm called *Relativistic Hamiltonian Monte Carlo (RHMC)* which is a variant of MCMC. In particular, we replace Newton's kinetic energy in the Hamiltonian with Einstein's relativistic kinetic energy, which makes the algorithm more robust. There are several extensions to RHMC, including a stochastic gradient version for scalability, a thermostat version based on the temperature of the physical system and a resulting optimization algorithm which gives comparable performance compared with the state-of-the-art.

Finally, we propose another sampling based inference method called the *Adaptive Importance Sampling with Exploration and Exploitation (Daisee)*, where we look into the problem of exploration-exploitation in adaptive importance sampling through establishing a natural connection between importance sampling and multi-armed bandit problem. In particular, through a finite-time regret analysis we show that the regret of the proposed algorithm grows sublinearly with time. Further, we propose a hierarchical extension of Daisee to encourage exploration in the region with high uncertainty.

The new models proposed in this thesis help to allow for more flexible Bayesian modelling and the inference techniques introduced can open new research directions for efficient and accurate posterior inference. These contribute to Bayesian inference and probabilistic machine learning.

Contents

List of Figures

Chapter 1

Introduction

This thesis takes the form of an integrated thesis, where each chapter contains an individual paper. This introduction serves as an overview of the topics that are included in this thesis, where detailed literature review are discussed later in each chapter.

Machine learning has become more and more popular over the decades and has been applied widely in different applications including robotics, healthcare, signal processing, speech and language, computer vision, recommender system, face recognition and the list goes on. In particular, deep learning [Goodfellow et al., 2016, Deng, 2011], which learns features of data with multiple levels of abstraction with non-linear operations [Deng et al., 2014], is a rich class of model to allow computers to learn complicated concepts and extract structured features.

Besides the rich representation of deep learning models, various well developed deep learning frameworks have further facilitated the active research in this area. These tools such as PyTorch [Paszke et al., 2017], Tensorflow [Abadi et al., 2016] and MxNet [Chen et al., 2015] apply automatic differentiation [Berland, 2006], have made it much easier for people to implement and train deep learning models.

However, despite the excitement of deep learning, it also has its limitations. For example, neural networks are often over-parametrized and can overfit the training data. They require a large amount of data and can be computationally expensive to deploy. Furthermore, sometimes it is hard to optimize deep neural networks [Goodfellow et al., 2014]. Deep learning is also opaque. There is no reasoning about the model and parameters which consequently, leads to a black-box that cannot be explained and raises doubts about reliability and biases.

Another main limitations of deep learning is that neural networks only use point estimates of the parameters and no calibrated uncertainties can be computed. More specifically, the parameters of neural networks are optimized and fixed to be the

values which maximise the likelihood of the observed data. This potentially can make it overly confident about making wrong predictions and worsen the generalisation performance. This thesis mainly tackles this limitation by finding ways of identifying and quantifying calibrated uncertainties, based on the framework of Bayesian methods. In particular, we propose

- a class of flexible models that are based on Gaussian processes (GP), which are a class of probabilistic models with tractable covariance structure. This method can be applied in a range of applications such as regression and collaborative filtering.
- a generative latent variable model that can be used for Bayesian optimization (BO). BO uses is a back-box optimization method that is based on GP, it is particularly useful when the objective function is expensive to evaluate and one can choose the locations at which the function can be evaluated. It takes into consideration of the uncertainties computed from the GP model when making decisions about the inputs to be evaluated at.
- a Hamiltonian Monte Carlo inference algorithm and its variants to better infer the distributions of unknown parameters.
- an adaptive important sampler with sub-linear finite-time regret that can be used for density estimation. In particular, it addresses exploration-exploitation trade off in the adaptive scheme.

1.1 Probabilistic Machine Learning

We now describe in more detail one of the key limitations of deep learning models: the uncertainty, which has motivated the research presented in this thesis. Why should we care about uncertainties? The world is full of uncertainties due to physical randomness, partial observability, incomplete knowledge, and accidents. It is important for a system to know what it does not know, and draw inference from noisy or ambiguous data, especially in applications such as healthcare and autonomous driving where a wrong decision can lead to devastating results. For example, a patient might have an illness with probability 40% and a deep learning model gives a prediction of not getting that disease. The patient may stop further diagnosis, however there is still 40% chance of getting that disease. Therefore we may prefer a model that gives calibrated uncertainties about a decision.

Probabilistic machine learning refers to the machine learning process using probabilistic models. It offers a framework for building systems that reason about uncertainty, have automatic model complexity control and structure learning. If test data is not seen during training, a probabilistic model can give higher uncertainty of the prediction. Moreover, confidence intervals or credible intervals of the parameters can also be deduced which is desirable in many aspects of machine learning and artificial intelligence such as forecasting, decision making, learning from noisy data, experimental design and the list goes on.

One of the most principled frameworks in probabilistic machine learning are Bayesian methods [Bishop, 2006, Berger, 2013] which combine prior knowledge with empirical evidence. There are many theoretical properties of Bayesian methods thanks to the rule of probability, which results in an elegant way of describing uncertainty, inferring the parameters given observed data and building credible intervals.

1.2 Bayesian Method

Suppose we have N training data and a **likelihood** model describing the distribution of the data with some unknown parameter θ. For labelled data, the data can comprise the training inputs $\{\mathbf{x}_j\}_{j=1}^N$ and responses $\{y_j\}_{j=1}^N$, where y can be continuous for regression problems or categorical for classification problems. For unlabelled data, the data can contain just $\mathbf{x}$ and the model $p(\mathbf{x}|\theta)$ describes the generative process of $\mathbf{x}$, with examples such as linear discriminant analysis or latent variable models. The unknown parameter θ is what we are interested in learning. It can be for example, the weight matrices in linear or logistic regression, or the weight matrices in neural networks.

Bayesian methods work by firstly placing some **prior** belief on the parameters' distributions $p(\theta)$, the resulting so called **posterior** distribution of the parameters is calculated by the Bayes rule. Taking Bayesian regression as an example, the posterior distribution can be computed through the conditional probability:

$$p(\theta|\mathbf{x}, y) = \frac{p(y|\mathbf{x}, \theta)p(\theta)}{p(y|\mathbf{x})}, \tag{1.1}$$

the **marginal** distribution is an integral of the data conditional on all possible parameters θ:

$$p(y|\mathbf{x}) = \int p(y|\mathbf{x}, \theta)p(\theta)d\theta, \tag{1.2}$$

given any test input $\mathbf{x}^*$, the predictive distribution of the response y^* can be calculated by

$$p(y^*|\mathbf{x}, y, \mathbf{x}^*) = \int p(y^*|\mathbf{x}^*, \theta)p(\theta|\mathbf{x}, y)d\theta. \tag{1.3}$$

1.2.1 Bayesian Modelling

One of the elegance of Bayesian method is the separation of modelling and inference. In other words, the form of the likelihood model $p(x|\theta)$ is chosen independently from the inference of θ. The likelihood and the prior distribution completely determine the posterior distribution, which is left to compute.

There is a wide choice of different models one can use to define the likelihood function. Historically, graphical models play an important role in Bayesian modelling. More recently, deep generative models have gained more attention in the machine learning community. We will discuss more about these in later sections.

1.2.2 Monte Carlo Sampling

After a model is selected, the next task is to learn the unknown parameters of the model. For Gaussian likelihood and Gaussian prior distributions, the posterior distribution of the parameters is analytical and can be calculated easily. However, there are many more complicated distributions where the integral in (1.2) is intractable and cannot be evaluated exactly. The predictive distribution in (1.3) therefore cannot be evaluated since it is an expectation with respect to the posterior distribution.

A popular inference technique that approximates an expectation is the Monte Carlo method, which turns an integration problem into a sampling problem. To ease notation, suppose the variable we are interested in learning the expectation with respect to is X. Given a test function ϕ, its expectation under a target density π can be estimated with the simple Monte Carlo estimate

$$\mathbb{E}[\phi(X)] = \int \phi(x)\pi(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\phi(x_i),$$

where $x_1, \cdots, x_N$ are drawn from π. However, it is not always straight forward to sample from π. For example, the predictive distribution in (1.3) is an expectation with respect to the posterior distribution $p(\theta|\mathbf{x}, y)$, which is the unknown target distribution. There has been an extensive literature of tackling this issue with the sampling approach. We hereby give a few popular examples.

1.2.2.1 Markov Chain Monte Carlo (MCMC)

MCMC works by constructing a Markov chain that has the desired distribution as its equilibrium distribution. There is a rich literature on different types of algorithms which sit in this category. For example, Metropolis-Hastings [Lu et al., 2018, Metropolis et al., 1953, Hastings, 1970] involves generating samples from a proposal distribution and accepting or rejecting it with certain probabilities, which is computed based on the target and proposal distributions; Gibbs sampling [Relaxation, 1984] iteratively updates the conditional distribution of each coordinate given the rest; Metropolis-within-Gibbs [Zeger and Karim, 1991] is a Gibbs sampler within which some of the component conditional distributions are sampled via Metropolis-Hasting; slice sampling [Neal, 2003] is based on the idea that any distribution can be sampled from by selecting uniformly spaced points under a probability distribution curve.

One of the attractive features of MCMC for sampling probability distributions is its better performance on higher dimensional problems. However, for very large dimensions, a density function will concentrate around its mode, but the volume over which we integrate that density is much larger away from the mode, meaning that the major contribution to any expectation comes from a nearly-singular neighbourhood. The guess-and-check strategy of random walk Metropolis is problematic since there are an exponential number of directions in which to guess, therefore a better exploration strategy is needed.

1.2.2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [Duane et al., 1987] tries to address this problem by exploiting information about the geometry of density though a Hamiltonian physical system. It is a variation of MCMC which differs from Metropolis-Hasting by reducing the correlation between successive sampled states.

The Hamiltonian is defined as

$$H(x, u) := U(x) + K(u), \tag{1.4}$$

where K and U represent the kinetic energy and potential energy respectively. The kinetic energy K is completely determined by the auxiliary parameter u, which is also called the momentum. The potential energy U contains information of the target distribution we wish to learn, which is defined as $U(x) := -\log \pi(x)$ for any target density π. The proposed updates of x and u are then determined from the Hamiltonian dynamics, which are then accepted or rejected in the Metropolis-Hasting procedure.

The use of Hamiltonian dynamics allows more efficient exploration in the parameter space, especially in the region that is far from the current position of the parameters.

1.2.2.3 Importance Sampling

Importance sampling (IS) is one of the historically important methods, which is a basic yet powerful key component of many advanced MC methods.

(Self normalised) IS works as follows, if the target density can be evaluated up to a constant, *i.e.*, if we have access to $f(x) := \frac{\pi(x)}{Z}$ where Z is the normalising constant, importance sampling [Kahn and Harris, 1951, Owen, 2013] works by sampling $x_1, \cdots, x_N$ from a known proposal distribution q. Define the importance weight to be $w(x) := \frac{f(x)}{q(x)}$, the expectation of the test function can be estimated by the (biased) estimate

$$\begin{aligned}\mathbb{E}[\phi(X)] &= \int \phi(x)\frac{\pi(x)}{q(x)}q(x)dx \\ &= \frac{1}{Z}\int \phi(x)\frac{f(x)}{q(x)}q(x)dx \\ &\approx \frac{\sum_{i=1}^N f(x_i)w(x_i)}{\sum_{i=1}^N w(x_i)},\end{aligned}$$

where the normalising constant can be estimated similarly:

$$Z = \int f(x)dx = \int \frac{f(x)}{q(x)}q(x)dx \approx \frac{1}{N}\sum_{i=1}^N w(x_i). \tag{1.5}$$

In the Bayesian framework, the target distribution π is the posterior distribution $p(\theta|x)$ where x and θ stand for the data and the parameters respectively. This is unknown, however, we have access to its unnormalised version $f(x) = p(\theta)p(x|\theta)$ with the normalising constant Z equal to the intractable marginal likelihood $p(x)$ using (1.4).

One of the important applications of IS where MCMC has troubles is rare event simulation [Owen and Zhou, 2000, Owen et al., 2017, Denny, 2001, Rubino and Tuffin, 2009]. Many practical problems can fall into this application such as bankruptcy in finance, energy grid failure [Yamangil et al., 2014]. Another application where MCMC does not work well is when the integrand is spiky or singular, such as in high energy physics [Aldins et al., 1970], graphical rendering [Kollig and Keller, 2006] and particle transport [Booth and Hendricks, 1984]. In contrast to other MCMC, IS also has the advantage of estimating the normalising constant Z which can be used in e.g. estimating model evidence etc.

1.2.2.4 Adaptive Importance Sampling

The performance of IS depends heavily on the choice of proposals. However, it is not easy to find a good proposal distribution a priori especially in high dimensions. Adaptive Importance Sampling (AIS) iteratively updates the proposal distribution q_t based on existing samples $x_1, \cdots x_t$. [Cornuet et al., 2012] used a mixture of present and past proposals and updated the mixing proportions based on the samples. [Martino et al., 2014] used a mixture with a fixed number of proposals per iteration with a user controllable computational cost. Cappé et al. [2008] used an entropy criterion to learn the weights and component parameters of a mixture importance sampling density. However, the finite-time analysis on the accuracy of the algorithms remains relatively less explored. In chapter 5, we propose an AIS algorithm which differs from previous work in the following perspectives: 1) we take into consideration of exploration and exploitation in the adaptive scheme; 2) we provide a finite-time regret analysis based on the KL divergence between the target and the proposal distributions, and show that the resulting algorithm has a sub linear regret. We show its promising performance both theoretically and empirically.

1.2.2.5 Limitations

In Monte Carlo methods, the efficiency of the global schemes such as importance sampling can depend on the proposal distributions which needs to be a good approximation to the target distribution. This is hard especially in high dimensional situations. For the local schemes such as Metropolis Hastings, it is hard to diagnose the convergence of the chains and it may exhibit slow mixing particularly in multi-modal scenarios.

1.2.3 Variational Inference

Another popular technique for approximate inference is variational inference (VI) [Beal et al., 2003, Ghahramani and Beal, 2000, Jordan et al., 1999] which approximates difficult densities through optimization.

The idea is to first choose a family of densities to approximate the target density, and then find the member of that family to best approximate the target, where the distance to the target density is measured by the Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951]. The KL is a standard way of measuring distance between two distributions and the KL between any two distributions p and q is defined as $KL(p||q) := \int p(x) \log \frac{p(x)}{q(x)} dx$. Interestingly, this method provides a lower bound of

the marginal likelihood $p(data)$, and minimizing the KL divergence is equivalent to maximising this lower bound.

In more detail, suppose we have observations $\mathbf{x}$ and latent variables $\mathbf{z}$, bayesian inference amounts to learning the posterior density $p(\mathbf{z}|\mathbf{x})$ given the likelihood $p(\mathbf{x}|\mathbf{z})$ and the prior $p(\mathbf{z})$

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}. \tag{1.6}$$

Note that we have slightly abused the use of notation and we are now interested in learning the posterior distribution of the latent variable $p(\mathbf{z}|\mathbf{x})$. The marginal likelihood $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz$, also called the *model evidence* is often intractable and expensive to calculate which makes inference in such models hard.

The first step in VI is to choose a family of distributions $\mathcal{Q}$ over the latent $\mathbf{z}$, where each member in the family $q(\mathbf{z})$ is a candidate approximation to the posterior $p(\mathbf{z}|\mathbf{x})$, the goal is to find the best candidate which is closest to the true conditional $p(\mathbf{z}|\mathbf{x})$ based on the KL measure:

$$q^*(\mathbf{z}) = argmin_{q(\mathbf{z})\in\mathcal{Q}} KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})). \tag{1.7}$$

However, computing this KL term involves calculating the marginal likelihood $p(\mathbf{x})$. Instead, note that

$$\log p(\mathbf{x}) = KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) - \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \tag{1.8}$$

$$= KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) + \mathcal{L}(q), \tag{1.9}$$

where $\mathcal{L}(\phi)$ is defined as the *evidence lower bound (ELBO)*:

$$\mathcal{L}(q) := -\int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z} \tag{1.10}$$

$$= \int q(\mathbf{z}) \log p(\mathbf{x}|\mathbf{z}) d\mathbf{z} - KL(q(\mathbf{z})||p(\mathbf{z})). \tag{1.11}$$

Since the KL is non-negative, we have that $\mathcal{L}(\phi) \leq \log p(x)$, meaning that maximising the ELBO is equivalent to miniming the KL distance between the approximation and the true posterior distribution of Z.

1.2.3.1 Mean Field Approximation

We now describe a simple family $\mathcal{Q}$ which is called the mean-field variational family [Peterson, 1987]. The key assumption is the independence between the latent variables

$$q(\mathbf{z}) = \prod_{j=1}^{D} q_j(z_j), \tag{1.12}$$

where D is the dimension of the latent variable. Each variational distribution q_j along the j-th coordinate can take on its own form, which is typically in the exponential family. Optimizing the ELBO in (1.10) can be done with e.g. Coordinate Ascent Variational Inference (CAVI) [Bishop, 2006]: due to the mean-field assumption, z_j are mutually independent, the optimal conditional distribution of z_j given the rest of the latent variables can be computed with

$$q_j^*(z_j) \propto \exp\{\mathbb{E}_{-j}[\log p(z_j|\mathbf{z}_{-j}, \mathbf{x})]\}. \tag{1.13}$$

CAVI works iteratively along different coordinate of $\mathbf{z}$, updating q_j for $j = 1, \cdots, D$ until convergence.

1.2.3.2 Stochastic Variational Inference

In the presence of big data, this method does not scale since each iteration involves going through the whole data set. A scalable variational inference method *Stochastic Variational Inference* [Hoffman et al., 2013] is proposed. It uses gradient based optimization based on natural gradient [Amari, 1998], combined with stochastic optimization [Robbins and Monro, 1985], which enables modern machine learning algorithms to scale massively.

The natural gradient takes account of the geometric structure of the probability space, and points to the direction that achieves maximum change in KL divergence. It can be computed by pre-multiplying the inverse Fisher information to the ordinary gradient in the exponential family [Amari, 1982].

On the other hand, stochastic optimization works by following an unbiased gradient which is a noisy and computationally cheap estimate for the true gradient. At each iteration, the update for the parameters is the stepsize ϵ_t multiplied by the unbiased estimate of the gradient. This has been proved successful provided that the stepsize sequence ϵ_t at each iteration t satisfy the condition [Robbins and Monro, 1985]:

$$\sum_t \epsilon_t = \infty, \qquad \sum_t \epsilon_t < \infty.$$

1.2.3.3 Amortised Methods

In the classic mean-field inference, we are often limited to exponential family distributions such as Gaussians. The parametric assumptions in VI are often too strong and unable to model posteriors that are complicated or multi-modal. The basic idea behind amortized inference is to use a powerful predictor $\mathbf{z} = q_\phi(\mathbf{x})$ to predict the optimal $\mathbf{z}$ based on the observations $\mathbf{x}$, where instead of approximating separate variables for each data point, the local variational parameters are replaced with a function whose parameters are shared across all the data points. Once this function is estimated, the latent variables can be computed by passing new data points through the function. When the function is parametrized by a neural network, this is called the inference network, which combines probabilistic modelling with the expressiveness of deep learning.

1.2.3.4 Limitations

In VI, the main limitation of mean field approximations is that they cannot capture correlations between different latent variables, which limits the expressibility of the variational family. Moreover, in contrast to MCMC methods, the statistical properties of variational inference are not yet well understood. It is also an interesting direction to develop variational inference methods which utilises other measures such as α-divergence in place of the KL divergence.

1.2.4 Gaussian Process

One of the well-known models that can be applied in Bayesian methods are *Gaussian Processes (GP)*, which we developed further in chapter 2. They are non-parametric and flexible models which have proved success in diverse applications ranging from Automatic Statistician [Duvenaud et al., 2013] to dynamic modelling of the environment in model-based reinforcement learning [Deisenroth and Rasmussen, 2011]. A Gaussian process is defined as a stochastic process, such that every finite collection of those random variables have a multivariate normal distribution [Rasmussen and Williams, 2005]. It is a distribution over a function where the 'parameters' θ refer to the function

itself. It is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$, also known as the kernel:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

For any finite number of collections $(x_1, \cdots, x_n)$, the joint distribution of $(f(x_1), \cdots, f(x_n))$ follow a multivariate distribution with mean vector $[m(x_1), \cdots, m(x_n)]^T$ and covariance matrix K whose (i, j)-th entry is equal to $k(x_i, x_j)$.

Suppose we have training data $(\mathbf{x}, \mathbf{y})$, denote the vector of the function evaluation by $\mathbf{f} := (f(x_1), \cdots, f(x_n))$. If we assume Gaussian likelihood with noise variance σ^2

$$p(\mathbf{y}|\mathbf{f}, \mathbf{x}) \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathrm{I}),$$

then the posterior distribution of the function f is also a Gaussian process

$$p(f|\mathbf{x}, \mathbf{y}) \sim \mathcal{GP}(m_{post}, k_{post}),$$

where

$$\begin{aligned} m_{post}(x) &= k(x, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma^2 \mathrm{I}]^{-1}\mathbf{y}, \\ k_{post}(x, x') &= k(x, x') - k(x, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma^2 \mathrm{I}]^{-1}k(\mathbf{x}, x), \end{aligned}$$

where we have assumed that the prior mean $m(x)$ is 0.

Typically, the mean function of a GP is set to be 0 and there are many choices of the covariance function such as rational quadratic, periodic, etc., with unknown hyper parameters to be learned.

The main problem of GP models is that inference costs $\mathcal{O}(N^3)$ where N is the number of training observations. Due to the inversion of the kernel matrix, it is computationally expensive and cannot scale to big data. There has been active research to make GP more scalable, such as the inducing point method [Snelson and Ghahramani, 2005, Titsias, 2009] which is based on a subset of the training points, also known as the pseudo inputs. [Wilson et al., 2014, Flaxman et al., 2015] exploited kronecker structure in the GP covariance matrix for high dimensional data, later they also extended it to combine the structural properties of deep learning architectures with the nonparametric flexibility of kernel methods [Wilson et al., 2016]. More recently, [Huggins et al., 2018] proposed an efficient variational approximate inference method based on the preconditioned Fisher (pF) divergence.

1.2.5 Bayesian Neural Networks

It is a natural idea to combine Bayesian methods with deep learning methods, in order to have a flexible deep learning system which takes uncertainty into consideration. One of the simplest models is Bayesian neural network. We first describe neural networks in detail.

Neural networks serve as the building blocks for deep learning, where the simplest neural network is a feed-forward neural network [Hornik et al., 1989]. Given data $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, a feed-forward neural network models non-linearity in the relationship of y given $\mathbf{x}$. It consists of hidden unit activations which are computed with a weight matrix W, a bias vector b and a (non-linear) activation function s:

$$h_{ik} = s\left(b_k^h + \sum_{j=1}^{p} W_{jk}^h x_{ij}\right).$$

Similarly for the output

$$\hat{y}_i = s\left(b^o + \sum_{k=1}^{m} W_k^o h_{ik}\right),$$

where W^h, b^h and W^o, b^o represent the weight and bias for the hidden unit and the output respectively.

Those parameters are usually learned by optimizing the log likelihood with gradient descent, which gives us a Maximum A Posterior(MAP) estimator. There has been a number of improvement on the optimization algorithms with adaptive learning rate such as Adam [Kingma and Ba, 2014b], RMSprop [Hinton et al.] and Adagrad [Duchi et al., 2011], etc.

What we have described above is often known as a feed-forward neural network or the dense layer, but there are many other neural network layers such as LSTM [Hochreiter and Schmidhuber, 1997, Gers et al., 1999] which is a recurrent neural network that models sequences of outputs, and convolutional neural network [LeCun et al., 1989, 1990, 1998] which is usually applied to visual images. Deep learning [LeCun et al., 2015, Goodfellow et al., 2016, Schmidhuber, 2015] stacks multiple layers of simpler neural networks which are able to model complicated structure with multiple levels of abstraction.

In Bayesian neural networks (BNN) [Hinton and Van Camp, 1993, MacKay, 1992, Neal, 1992, Peterson, 1987], the parameters of the neural networks such as the weight matrices are deemed as random variables. Rather than a point estimate, a distribution

over the weights is learned with Bayesian inference. For ease of notation, we embed the parameter b into the weight matrix W, a prior distribution $p(W)$ is placed over the weight, and its posterior distribution $p(W|data)$ is computed which usually is not tractable and difficult to evaluate. Instead, one can use approximate variational inference in section 1.2.3 or MCMC in section 1.2.2 to approximate the posterior distribution.

1.2.6 Deep Generative Models

Generative models are powerful unsupervised learning method which can represent complicated data distributions. The goal is to learn the true data distribution and be able to generate new data from the true distribution. It leverages the flexibility of neural networks to learn a function which can approximate the true distribution.

Two of the most commonly used and efficient approaches are Variational Auto-encoders (VAE) [Kingma and Welling, 2013, Rezende et al., 2014] and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]. VAE maximises the marginal distribution of the data with variational inference and GAN aims at achieving an equilibrium between Generator and Discriminator, where the Generator generates samples and tries its best to fool the Discriminator, and the Discriminator tries to distinguish between the true samples from the generated ones.

We will discuss in more detail about VAE which serves as the building blcok for our work in chapter 3. .

1.2.6.1 Variational Auto-encoder

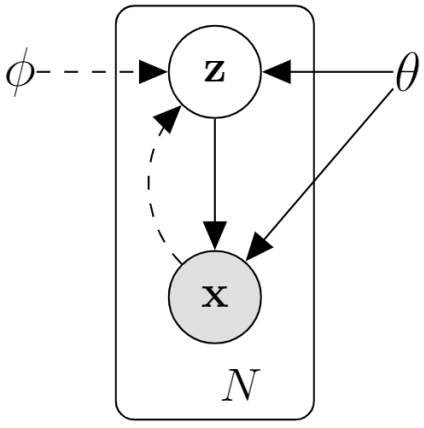


Figure 1.1: Graphical representation of variational auto-encoder.

The graphical representation of VAE is displayed in Figure 1.1. It assumes a Gaussian prior over $\mathbf{z}$ with $p(\mathbf{z}) = \mathcal{N}(0, I)$ and it employs amortised mean-field approximation of the posterior distribution of $\mathbf{z}$, also known as the *recognition network* or *encoder*:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{j=1}^{N} q_\phi(\mathbf{z}_j|\mathbf{x}_j), \tag{1.14}$$

where N is the number of data points. Typically, $q_\phi(\mathbf{z}|\mathbf{x}_j)$ is chosen to be Gaussian:

$$q_\phi(\mathbf{z}_j|\mathbf{x}_j) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}_j), \sigma^2_\phi(\mathbf{x}_j)\mathbf{I}), \tag{1.15}$$

where $\mu_\phi(\mathbf{x}_j)$ and $\sigma^2_\phi(\mathbf{x}_j)$ are outputs from neural networks parametrized by ϕ. Similarly, the *generative network*, also known as the *decoder* is also typically defined as Gaussian:

$$q_\theta(\mathbf{x}_j|\mathbf{z}) = \mathcal{N}(\mathbf{x}_j|\mu_\theta(\mathbf{z}), \sigma^2_\theta(\mathbf{z})\mathbf{I}), \tag{1.16}$$

where $\mu_\theta(\mathbf{z})$ and $\sigma^2_\theta(\mathbf{z})$ are outputs from neural networks parametrized by θ. Recall (1.10), the ELBO of a single data $\mathbf{x}_j$ is computed as

$$\mathcal{L}(\theta, \phi; \mathbf{x}_j) = -KL(q_\phi(\mathbf{z}|\mathbf{x}_j)||p(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}_j|\mathbf{x}_j)}[\log p_\theta(\mathbf{x}_j|\mathbf{z})]. \tag{1.17}$$

The KL term is tractable and can be computed analytically. The variational parameters ϕ and generative parameters θ can be learned by optimizing $\mathcal{L}$ with e.g. gradient descent. The differentiation of $\mathcal{L}$ with respect to θ is straight forward and can be estimated with Monte Carlo estimates as we explained in section 1.2.2. However, the gradient with respect to ϕ is problematic since the expectation in (1.16) is with respect to q_ϕ which depends on ϕ:

$$\begin{aligned}
\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{x}_j|\mathbf{z})] &= \int \nabla_\phi q_\phi(\mathbf{z}) \log p_\theta(\mathbf{x}_j|\mathbf{z}) dz \\
&= \int q_\phi(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z}) \log p_\theta(\mathbf{x}_j|\mathbf{z}) dz \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{x}_j|\mathbf{z}) \nabla_{q_\phi(z)} \log q_\phi(z)] \\
&\approx \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(\mathbf{x}_j|\mathbf{z}^{(l)}) \nabla_{q_\phi(\mathbf{z}^{(l)})} \log q_\phi(\mathbf{z}^{(l)}),
\end{aligned}$$

where $z^{(l)}$ are draws from the approximate posterior $q_\theta(\mathbf{z}|\mathbf{x}_j)$ which depend on ϕ, and we have used score identity [Cox and Hinkley, 1974] in the second equality. It is shown that this gradient estimator exhibits very high variance Paisley et al. [2012]. One of the key contributions of VAE is the reparametrization trick which reduces the variance and makes this method practically usable. The idea is to reparametrize $\mathbf{z}$ with a differentiable transformation $g_\phi(\epsilon, \mathbf{x}_j)$ of an auxiliary noise variable ϵ:

$$\mathbf{z} = g_\phi(\epsilon, \mathbf{x}_j), \qquad \epsilon \sim p(\epsilon),$$

where the expectation can be written as

$$\mathbb{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{x}_j|\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{x}_j|g_\phi(\epsilon, \mathbf{x}_j))] \approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}_j|g_\phi(\epsilon^{(l)}, \mathbf{x}_j)),$$

where $\epsilon^{(l)} \sim p(\epsilon)$. One can then apply stochastic optimization by selecting a mini batch of data to form an unbiased stochastic gradient estimate.

1.2.6.2 Advances in VAE

VAE opens a new array of research in probabilistic deep models and there have been a number of extensions based on it: Sohn et al. [2015] introduced a latent variable and developed a deep conditional generative model for structured output prediction. Higgins et al. [2016] also introduced a latent representation from raw image data in an unsupervised manner. Kingma et al. [2016] factorizes over dimensions in VAE by modeling dependencies between pixels within an image using a conditional model, where the dimensions are generated in a sequential manner. Johnson et al. [2016] used a structured VAE which is a rich latent representation, whose recognition networks output conjugate graphical model potentials rather than the non-conjugate observation likelihoods. Figurnov et al. [2018] proposed another re-parametrization trick that works for a broader range of distributions. More recently, Tucker et al. [2017] used a control variate to reduce the variance in the model; Tucker et al. [2018] proposed an unbiased gradient estimator through estimating the bias efficiently with the re-parametrization trick.

The mean-field method and simple structured approximations such as Gaussian can be restrictive. In order to have efficient and rich posterior approximation, Rezende and Mohamed [2015] proposed flexible, arbitrarily complex and scalable approximate posterior distributions, which are constructed through a normalizing flow. The idea is to transform a simple initial density into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained.

1.3 Case study

Deep learning is flexible to model non-linearity in the data structure, and Bayesian methods are well established methods that calibrate uncertainties. Many research has gone into the direction of the combination of both, we give two case studies of the application of Bayesian deep learning which we will discuss in later chapters.

1.3.1 AutoML

Much effort has been put into automating machine learning. One such example is the Automatic Statistician (AS) [Duvenaud et al., 2013], where given any data set, a model in the GP class is selected and learned, and an interpretable report is generated automatically. Another example is probabilistic programming [Chen et al., 2014b]. It is a universal inference engine which does inference automatically given input data.

1.3.2 Bayesian Optimization

Bayesian optimization [Hernández-Lobato et al., 2014] has a similar flavour to active learning. The goal of BO is to optimize some objective function which is possibly expensive to evaluate. The idea is to view it as a sequential decision-making problem and pick informative points to be evaluated in order to find the optimum. It is based on GP models and the acquisition function typically relies on the variance output from the GP model prediction. We will discuss it in more detail in chapter 3.

1.4 Challenges

Despite the promising results with Bayesian methods, there are also several challenges and interesting open research directions.

1.4.1 Expensive Computation

Compared with deep learning, Bayesian machine learning requires more storage and is computationally more expensive. This is due to the computation of the posterior distribution which also involves calculating the likely intractable marginal likelihood. When the neural network architectures are wide and deep, the large number of parameters make it impractical to scale.

There has been a number of attempts to scale up Bayesian deep learning. For example, Khan and Nielsen [2018] proposed new natural-gradient algorithms to reduce

computational costs for Gaussian mean-field VI, by perturbing the network weights during gradient evaluations within the Adam optimizer. Mishkin et al. [2018] also took advantage of natural gradients to estimate a 'diagonal plus low-rank' structure of the covariance which gave faster and more accurate estimation of uncertainty empirically. Kronecker-factored Approximate Curvature (K-FAC) based methods Martens and Grosse [2015], Grosse and Martens [2016] approximate the Fisher information matrix of a neural network, resulting in a much faster algorithm compared with stochastic gradient descent with momentum. However, there is still a lot of room to develop scalable Bayesian inference techniques.

1.4.2 Flexible Models

A mis-specified model leads to incorrect learning. Another question is how to design flexible models that are capable of modelling complex data structure. Up to now, the form of the likelihood or prior distribution is still restrictive. Non-parametric models such as Gaussian Processes still assume Gaussian prior. Deep generative models typically assume Gaussian distributions for the prior of the latent variable and the likelihoods, where the mean and variance are outputs from neural networks.

Recently, Garnelo et al. [2018b,a] proposed Neural Processes (NP), which followed the idea of VAE and combined neural networks with Gaussian processes. It is capable of rapid adaptation to new observations and estimating the uncertainty in their predictions.

1.4.3 Accurate and Efficient Inference

Most of the time, the posterior is intractable and approximations are needed. However, there is no single posterior inference method that works well across all Bayesian models. How to develop an inference method that is scalable, accurate, effective and can be adapted to more domains remains an interesting challenge.

The interface between variational inference and MCMC is less explored. Salimans et al. [2015] showed some first results bridging the gap between the two. It incorporated one or more steps of MCMC into the variational approximation, resulting in fast posterior approximation through the maximization of an explicit objective. A thorough analysis of which approximate inference method to use and how to take advantage of both is a promising area of research.

1.5 Summary

So far in this chapter we have specified the necessity of probabilistic machine learning and the importance of calculating calibrated uncertainties. We have also reviewed state-of-the-art modelling inference techniques in Baysian inference. There are a lot of open questions in Bayesian machine learning, such as developing computationally efficient and easy to implement probabilistic models, improving interpretability in Bayesian deep learning, theoretical analysis of the error in different approximate inference. Making Bayesian deep learning more efficient, accurate and automatic remains an important and exciting research area.

1.6 Thesis Outline

This thesis consists of four main chapters.

We discuss Tucker Gaussian Process in chapter 2. This is joint work with Hyunjik Kim, Seth Flaxman and Yee Whye Teh. It is based on the paper

- **Xiaoyu Lu**, Hyunjik Kim, Seth Flaxman and Yee Whye Teh
  Tucker Gaussian Process for Regression and Collaborative Filtering
  https://arxiv.org/abs/1605.07025
  Women in Machine Learning Workshop, NIPS, 2016

The main contributions I have made to this work includes:1) the formalization of the model; 2) the derivation of the SGD algorithm; 3) experimental results on the Irish wind data set; 4) partial experimental results on the MovieLens dataset.

The first component of the thesis is to build a Tucker Gaussian Process that reduces computational cost, and in particular, gives superior performance when the input space is structured and separable. Another stunning point of TGP is that it can naturally incorporate side information of users and items in the application of collaborative filtering.

In chapter 3, we build a probabilistic model using Gaussian Processes as building blocks in the context of optimization. This is joint work with Javier González, Zhenwen Dai and Neil Lawrence and I was the main contributor to the project.

- **Xiaoyu Lu**, Javier González, Zhenwen Dai and Neil Lawrence
  Structured Variationally Auto-encoded Optimization
  http://proceedings.mlr.press/v80/lu18c.html
  International Conference on Machine Learning (ICML) 2018.

We call our model *Structured Variational Optimization (SVO)*, which does Bayesian optimization over structured input space. In Science and Engineering, optimization problems with very structured input domains are ubiquitous.

Take *Automatic Statistician (*AS*)* as an example. With a GP as the class of models of choice, the goal is to automatically select the best kernel combination to explain a data set, which is chosen by enumerating a countably infinite set of arbitrarily complex kernels composed via additions and multiplication of simple ones [Duvenaud et al., 2013]. Interestingly, the final goal of the AS is not just to fit a model to data, but to write a report that uses the type of combination as the main element to interpret the data.

However, the space of the models is infinite and combinatorial, optimization over this space is challenging. The idea is to learn a low dimensional latent space which is continuous and Euclidean using a VAE, and conduct Bayesian optimization on the low dimensional space instead. We demonstrate experimentally that our method gives competitive results on a number of datasets.

Chapter 4 talks about a variant of Hamiltonian Monte Carlo, namely the *Relativistic Hamiltonian Monte Carlo (RHMC)* algorithm. This is joint work with Valerio Perrone, Leonard Hasenclever, Sebastian J Vollmer and Yee Whye Teh based on the publication.

- **Xiaoyu Lu**, Valerio Perrone, Leonard Hasenclever, Sebastian J Vollmer and Yee Whye Teh.
  Relativistic Monte Carlo
  https://arxiv.org/abs/1609.04388.
  International Conference on Artificial Intelligence and Statistics, 2017.

My main contribution to this project includes: 1) the derivation and implementation of the RSGD algorithm. 2) the derivation and implementation of the RSGNHT algorithm; 3) experimental results on Banana, logistic regression and MNIST examples.

The Relativistic Hamiltonian Monte Carlo (RHMC) algorithm, which is based on continuous-time physical systems that allow the efficient simulation of posterior distributions, are an important mainstay of Bayesian machine learning and statistics. Hamiltonian Monte Carlo (HMC) [Duane et al., 1987, Neal, 2010, Carpenter, 2015] is based on Newtonian dynamics on a frictionless surface, and has been argued to be more efficient than techniques based on diffusion [Xifara et al., 2014]. On the other hand, stochastic gradient MCMC techniques based on diffusive dynamics [Welling and Teh, 2011a, Ding et al., 2014, Chen et al., 2014a] have allowed scalable Bayesian learning using mini-batches. An important consideration when designing such MCMC

algorithms is adaptation or tuning to the geometry of the space under consideration [Girolami and Calderhead, 2011, Beskos et al., 2013, Patterson and Teh, 2013].

We are interested in making MCMC algorithms based on physical simulations more robust by directly controlling the velocity of the particle. This is achieved by replacing Newtonian dynamics in HMC with relativistic dynamics [Einstein, 1905], whereby particles cannot travel faster than the "speed of light". We also developed relativistic variants of stochastic gradient MCMC algorithms, including stochastic gradient Langevin dynamics, SGHMC and Nosé-Hoover thermostat, and show that they give promising results and are more robust than the classical Newtonian variants.

Finally, we discuss exploration and exploitation in *Adaptive Importance Sampling (AdaIS)* in chapter 5. This is work supervised by Yee Whye Teh and I am the main contributor of the project. This leads to the paper

- **Xiaoyu Lu**, Tom Rainforth, Yuan Zhou, Jan-Willem van de Meent and Yee Whye Teh.
  On Exploration, Exploitation and Learning in Adaptive Importance Sampling.
  https://arxiv.org/abs/1810.13296
  arXiv preprint arXiv:1810.13296 (2018)

While Markov Chain Monte Carlo and sequential Monte Carlo are the two most common families of Monte Carlo methods, in recent years there has been growing interest in various forms of Adaptive Importance Sampling (AIS) [Liu, 2008, Cappé et al., 2004, Cornuet et al., 2012, Cappé et al., 2008], due to their simplicity and ability to estimate normalization constants of interest. In Chapter 5, we initiate the study of adaptive Monte Carlo as online learning procedures, and argue for the importance of the trade-off between exploration and exploitation in adaptive Monte Carlo. We study this issue in the context of adaptive importance sampling (AIS), where we borrow ideas around upper confidence bounds in the bandit literature to construct a partition-based AIS with $O(\sqrt{T}(\log T)^{\frac{3}{2}})$ cumulative regret, where T is the number of iterations. We further extend our AIS algorithm to use an adaptively constructed recursive partition of the sample space, which is a non-parametric extension of the model.

Chapter 2

Tucker Gaussian Process

- **Xiaoyu Lu**, Hyunjik Kim, Seth Flaxman and Yee Whye Teh,
  Tucker Gaussian Process for Regression and Collaborative Filtering,
  https://arxiv.org/abs/1605.07025,
  Women in Machine Learning Workshop, NIPS, 2016.

My contribution to the paper includes:

- The formalization of the model
- The derivation of the SGD algorithm
- Experiments on the Irish wind data set
- Partial experiments on the MovieLens data set.

.

2.1 Abstract

We tackle the problem of collaborative filtering (CF) with side information, through the lens of Gaussian Process (GP) regression. Driven by the idea of using the kernel to explicitly model user-item similarities, we formulate the GP in a way that allows the incorporation of low-rank matrix factorisation, arriving at our model, the *Tucker Gaussian Process* (TGP). Consequently, TGP generalises classical Bayesian matrix factorisation models, and goes beyond them to give a natural and elegant method for incorporating side information, giving enhanced predictive performance for CF problems. Moreover we show that it is a novel model for regression, especially well-suited to grid-structured data and problems where the dependence on covariates is close to being separable, meaning that the response can be roughly expressed as (a sum of) the product of functions over each covariate.

2.2 Introduction

Collaborative filtering (CF) defines a branch of techniques for tackling the following supervised learning problem: making predictions (filtering) about the preferences of a user, based on information regarding the preferences of many users (collaboration). We are given data in the form of a partially observed rating matrix R, where R_{ij} is the rating of user u_i on movie v_j for $i = 1, \ldots, n_1, j = 1, \ldots, n_2$. CF aims to predict missing entries of R by only using the observed entries. Hitherto, matrix factorisation approaches [Billsus and Pazzani, 1998, Koren, 2009, Piotte and Chabbert, 2009, Töscher et al., 2009] have been the basis for many successful CF models. These model R as a product of two low rank matrices $R \approx UV^\top$, hence $R_{ij} \approx \sum_k U_{ik}V_{jk}$. On the other hand, content-based filtering predicts user ratings based on attributes of users (e.g. age, sex) and items (e.g. genre). CF with side information is a combination of the two, aiming to predict user ratings using both ratings data and user/item attributes.

There has been a wide range of work on CF with side information, mostly building on the framework of matrix factorisation. Suppose user/item side information is given in the form of feature matrices $F = [\omega(u_1), ..., \omega(u_{n_1})]^\top \in \mathbb{R}^{n_1 \times r}$ and $G = [\omega'(v_1), ..., \omega'(v_{n_2})]^\top \in \mathbb{R}^{n_2 \times r}$. *Matrix co-factorization* [Singh and Gordon, 2008] attempt to factorise F, G and R simultaneously, whereas the *Regression-based Latent Factor Model* [Agarwal and Chen, 2009] assumes instead that U and V are linear in F and G. *Bayesian Matrix Factorization with Side Information* (BMFSI) [Porteous and Welling, 2010] gives an additive model in the sense that R is assumed to be the sum of the standard matrix factorisation prediction $UV^\top$ and linear contributions of F and G. *Hierarchical Bayesian Matrix Factorization with Side Information* [Park et al., 2013] is an extension of BFMSI with Gaussian-Wishart hyperpriors on the prior mean and variance of U and V.

Gaussian Processes (GPs) are a popular class of Bayesian nonparametric priors over functions [Rasmussen and Williams, 2005], and have served as flexible models across a range of machine learning tasks, e.g. regression, classification, dimensionality reduction [Lawrence, 2004] and CF [Lawrence and Urtasun, 2009, Yu et al., 2006]. In a regression setting with input and output pairs, a key advantage of GPs is that we can use the kernel to explicitly model similarity in the outputs between a pair of input values. We use this to model similarity between users/items given side information, forming the outset of the paper. We model the ratings as $R_{ij} \sim \mathcal{N}(f(u_i, v_j), \sigma^2)$ and $f \sim \mathcal{GP}(k_1 \times k_2)$ where kernels k_1 and k_2 model user and item similarities

respectively. However, a direct GP regression application is infeasible due to its $O(N^3)$ computational cost, because for most CF problems, the number of ratings N ranges from a hundred thousand to hundreds of millions. Low-rank matrix factorisation remedies this problem, and also underlies the state of the art approaches for CF. Hence it is natural to look for a connection between GPs and low-rank matrix factorisation, which is the motivation and contribution of our work.

To develop a framework for this connection, we first propose a novel approximation scheme for GPs. Our starting point is the Kronecker structure that arises naturally when working with kernels that are products of simpler constituent kernels (say each dependent on one covariate dimension). Coupled with the weight space view of GPs, we can represent a draw from the GP as a product between a random weight tensor and a collection of feature vectors (one for each constituent kernel). The weight tensor can be very large for high dimensional problems, and our proposal is to approximate it using a low-rank Tucker decomposition [Tucker, 1966] instead. This reduces the effective number of parameters that need to be learnt, and forms the link between GPs and matrix factorisation methods in CF. Thus we arrive at our model, the Tucker Gaussian Process (TGP).

We make the following contributions:

- TGP is an elegant and effective method for modelling user/item similarities via kernels to exploit side-information. As far as we know, ours is the first work to use GPs for modelling similarities via side information, with explicit correspondence between similarities and the kernel.
- TGP generalises classical Bayesian Matrix factorisation models [Salakhutdinov and Mnih, 2008b,a], bridging the gap between matrix factorisation methods and GP methods in CF.
- Sub-linear scaling of TGP, achieved by stochastic gradient descent, makes it suitable for CF problems that typically have large data sets infeasible for GPs.
- TGP is applicable to certain regression problems where the regression function is separable in the covariates. We reason that the Tucker decomposition acts as a regulariser for the GP that helps control overfitting, and verify that TGP outperforms GPs in generalisation performance for these problems.

Outline of chapter We formulate TGP in the general regression setting in Section 2.3, and describe its central application to CF with side information in Section 2.4,

followed by related work and discussion in Section 2.5. We present experimental results in Section 2.6 and conclude in Section 2.7.

2.3 Tucker Gaussian Process Regression

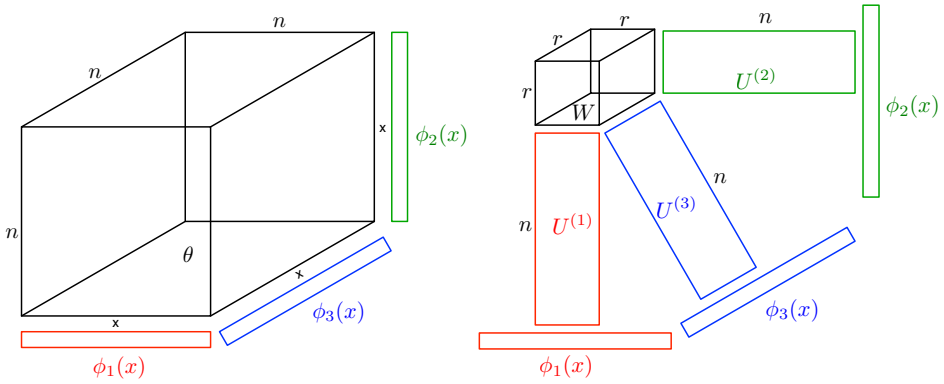
2.3.1 Tucker GP Regression

Consider a regression problem with inputs $x_1, \ldots, x_N \in \mathcal{X}$ and corresponding observations $y_1, \ldots y_N \in \mathbb{R}$. We assume $y_i | x_i \sim \mathcal{N}(f(x_i), \sigma^2)$ for some $f : \mathcal{X} \to \mathbb{R}$ and that the observations are independent. The aim is to learn f. One approach is to put a Gaussian Process (GP) prior on f, with zero mean and covariance k. The training then consists of computing the posterior GP. The problem of using this in a CF setting is that training costs $O(N^3)$ operations. N, the number of ratings, usually ranges from a hundred thousand to hundreds of millions in CF, making inference infeasible.

The weight space view of GPs offers a natural way of dealing with the problem: suppose there exists a feature map $\phi : \mathcal{X} \to \mathbb{R}^n$ (where n is the number of features) such that $k(x, x') = \phi(x)^\top \phi(x') \; \forall x, x' \in \mathcal{X}$. Then the GP is equivalent to Bayesian Linear Regression with feature vectors used for each row of the design matrix [Rasmussen and Williams, 2005]:

$$\begin{aligned} y | x, \theta &\overset{ind}{\sim} \mathcal{N}(f(x), \sigma^2), \quad f(x) = \theta^\top \phi(x) \\ \theta &\sim \mathcal{N}(0, I), \quad \theta \in \mathbb{R}^n \end{aligned} \tag{2.1}$$

Now training takes $O(Nn^2)$ time, and is scalable for $n \ll N$.



(a) Tensor (b) Tucker

Figure 2.1: Tensor & Tucker Decomposition representation of regression function for $D = 3$.

Consider the case of product kernels, where the kernel can be written as follows:

$$k(x_i, x_j) = \prod_{d=1}^{D} k_d(x_i, x_j) \tag{2.2}$$

and suppose there are feature maps $\phi_d : \mathcal{X} \to \mathbb{R}^n$ such that $k_d(x_i, x_j) = \phi_d(x_i)^\top \phi_d(x_j)$. Then we can write $k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$ where $\phi(x) = \otimes_{d=1}^{D} \phi_d(x)$ is the Kronecker product of the ϕ_d. For simplicity, we assume that the features $\phi_d(x)$ have the same dimension n across $d = 1, \cdots, D$. Returning to (2.1):

$$f(x) = \theta^\top \phi(x) = \theta^\top \Big(\otimes_{d=1}^{D} \phi_d(x) \Big) = \theta \times_{d=1}^{D} \phi_d(x) \tag{2.3}$$

where θ has been reshaped as a D-dimensional tensor in $\mathbb{R}^{n \times \cdots \times n}$ in the rightmost expression, as in Figure 2.1a. We define the tensor product notation as follows:

$$\theta \times_{d=1}^{D} \phi_d := \; vec(\theta)^\top \otimes_{d=1}^{D} \phi_d = \sum_{i_1, \ldots, i_D = 1}^{n} \theta_{i_1, \ldots, i_D} \prod_{d=1}^{D} (\phi_d)_{i_d}.$$

We refer to (2.3) as the *full-rank* model, and use θ as a tensor for the rest of the paper.

This *full-rank* model is problematic in high dimensions: the size of θ grows as n^D, so the function computation become infeasible. In two dimensions, a low rank approximation can be done with singular value decomposition, however, it is more complicated in higher dimensions. There exist multiple notions of tensor decomposition, such as PARAFAC [Bro, 1997] and Tucker [Tucker, 1966] decomposition. Here we introduce the novel *Tucker Gaussian Process* (TGP) model, where we approximate θ using a low-rank Tucker decomposition. The Tucker decomposition is defined as a tensor-matrix product between a low rank core tensor $W \in \mathbb{R}^{r \times \cdots \times r}$ of dimension D and matrices $U^{(1)}, \ldots U^{(D)} \in \mathbb{R}^{n \times r}$, as in Figure 2.1a. The $(i_1, \ldots, i_D)^{th}$ entry of θ is defined as

$$\theta_{i_1, \ldots, i_D} := \sum_{l_1, \cdots, l_D} W_{l_1, \ldots, l_D} \prod_{d=1}^{D} U^{(d)}_{i_d l_d}. \tag{2.4}$$

We denote it as $\theta \approx W \times_{d=1}^{D} U^{(d)\top}$, and hence the $(i_1, \ldots, i_D)^{th}$ entry is $W \times_{d=1}^{D} U^{(d)}_{i_d}$ with $U^{(d)}_{i_d}$ a column vector representing the i_d^{th} row of $U^{(d)}$, n is the number of features in each dimension and r is the rank. Note that we are free to use a different n and r for each dimension, but assume these are the same for convenience of notation.

We must also place suitable priors on W and $U^{(d)}$ to match the iid $\mathcal{N}(0, 1)$ prior on each entry of θ. We place iid priors $\mathcal{N}(0, 1)$ on each entry of W, and $\mathcal{N}(0, \sigma_u^2)$ on

each entry of $U^{(d)}$. Setting $\sigma_u^2 = \frac{1}{r}$, we match the first two moments of $W \times_{d=1}^{D} U^{(d)\top}$ and θ. We then prove in Appendix 2.8.1 that each entry of $W \times_{d=1}^{D} U^{(d)\top}$ converges in distribution to $\mathcal{N}(0,1)$ as $r \to \infty$.

In summary our TGP regression model approximating data from a GP with product kernel (2.2) and homoscedastic noise is:

$$y|x \overset{ind}{\sim} \mathcal{N}(f(x), \sigma^2),\ f(x) = W \times_{d=1}^{D} \left(U^{(d)\top}\phi_d(x)\right), \tag{2.5}$$

where $\phi_d : \mathcal{X} \to \mathbb{R}^n$ are feature maps such that $k_d(x_i, x_j) = \phi_d(x_i)^\top \phi_d(x_j)$, and we have iid $\mathcal{N}(0,1)$ and $\mathcal{N}(0, \frac{1}{r})$ priors on the entries of W and $U^{(d)}$ respectively. Note that the resulting model is no longer a GP and its marginal likelihood is not tractable. We detail the learning procedure in section 2.3.3.

2.3.2 Choice of Feature Map

So far, we have assumed that the kernels k_d can be written as the inner product of feature vectors: $k_d(x_i, x_j) = \phi_d(x_i)^\top \phi_d(x_j)$. We investigate the situations where this assumption holds. When this doesn't hold, we explore other choices of ϕ that approximate k_d.

Identity features One case where we can write kernels as inner products of features is with identity kernels $k_d(x_i, x_j) = \delta_{ij}$. This is particularly convenient in matrix factorisation as we will explain later. The features are unit vectors: $\phi_d(x_i) = e_i := (0, \cdots, 0, 1, 0, \cdots)^\top$ with the non-zero at the i^{th} entry, hence $U^{(d)\top}\phi_d(x_i) = U_i^{(d)}$. However this implies $U^{(d)} \in \mathbb{R}^{N \times r}$ (or $\mathbb{R}^{n_d \times r}$ for inputs on a grid where n_d denotes the number of data points in each dimension, *i.e.*, $N = \prod_{d=1}^{D} n_d$), so for N (or n_d) too big, computations can become too costly both in time and memory. A workaround is to use feature hashing [Weinberger et al., 2009] to obtain shorter features whose inner products are unbiased estimates of inner products of the original features. This technique can be applied to arbitrary features where the number of features is too large. See Appendix 2.8.2 for details.

We can also deal with cases where the data lies on a grid using **Cholesky features**, or in the most general case where the data doesn't lie on a grid and k_d cannot be expressed as the inner product of finite feature vectors using **Random Fourier features**. See Appendix 2.8.4 for details.

2.3.3 Learning

In TGP we would like to learn the posterior distribution of U and W. The simplest and fastest method of learning is Maximum a Posteriori (MAP), whereby we approximate

the posterior with point estimates $\hat{U}, \hat{W} = \arg\max_{U,W} p(U, W|y)$. For the optimisation we may use stochastic gradient descent (SGD) to approximate the full gradient, for which we get a time complexity of $O(m(nrD + r^D D))$ operations for computing the stochastic gradient on a mini-batch of size m, which is linear in N.

The problem with a MAP estimate for U, W is that only the posterior mode is used, and the uncertainty encoded in the shape of the posterior distribution is ignored. In a Bayesian setting, we wish to use samples from the posterior and average predictions over samples. For data where we can afford an $O(N)$ runtime, we may use sampling algorithms such as Hamiltonian Monte Carlo (HMC) [Duane et al., 1987, Neal, 2011]. The runtime for each HMC leapfrog step is $O(N(nrD + r^D D))$, the same time complexity as a step of full-batch gradient descent.

2.4 TGP for Collaborative Filtering with Side Information

In order to apply TGP to CF, let us first formulate the problem using GPs. It is natural to model this as a supervised regression problem with $R_{ij} \sim \mathcal{N}(f(u_i, v_j), \sigma^2)$ and prior $f \sim \mathcal{GP}(0, k)$ [Yu et al., 2006]. Note that this is particularly suitable with side information, since kernels can be interpreted as measures of similarity; we can design k to encode similarities between users/movies given by the side information. Hence we may further exploit the use of GPs for addressing this problem. In particular we use a product kernel $k((u_i, v_j), (u_{i'}, v_{j'})) = k_1(u_i, u_{i'})k_2(v_j, v_{j'})$ since we expect similar ratings for two user/movie pairs if the users are similar *and* the movies are similar. When there is no side information, it is sensible to use identity kernels $k_1(u_i, u_{i'}) = \delta_{u_i u_{i'}}$, $k_2(v_j, v_{j'}) = \delta_{v_j v_{j'}}$. i.e. that distinct users and movies are not similar a priori. With side information, we may add on further kernels κ_1, κ_2 modelling similarity between users/movies: $k_1(u_i, u_{i'}) = a_1^2\delta_{u_i u_{i'}} + b_1^2\kappa_1(u_i, u_i')$, $k_2(v_j, v_{j'}) = a_2^2\delta_{v_j v_{j'}} + b_2^2\kappa_2(v_j, v_j')$, where a and b are parameters controlling the extent to which similarity in side information leads to similarity in preference.

However, it is not immediately clear how this single GP framework relates to the matrix factorisation approach. We show that our proposed TGP forms a natural connection between these two approaches, and that we recover classic matrix factorisation models as a special case. To apply TGP, first note that we have $D = 2$, and the Tucker Decomposition is simply a low-rank matrix factorisation. Using the notation U, V instead of $U^{(1)}, U^{(2)}$, we have that $\theta \approx UWV^\top$, hence

$f(u_i, v_j) = \phi_1(u_i)^\top UW(\phi_2(v_j)^\top V)^\top$. With the identity kernel, we have unit vector features $\phi_1(u_i) = e_i \in \mathbb{R}^{n_1}$ and $\phi_2(v_j) = e_j \in \mathbb{R}^{n_2}$. TGP therefore simplifies to:

$$R_{ij} \overset{ind}{\sim} \mathcal{N}(f(u_i, v_j), \sigma^2), \quad f(u_i, v_j) = U_i^\top W V_j, \tag{2.6}$$

with iid $\mathcal{N}(0, \sigma_u^2)$ priors on each entry of U, V where U_i, V_j are column vectors representing the i^{th} and j^{th} row of U and V respectively. Note that with $W = I$ fixed, we recover Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2008a], a particularly effective Bayesian model in the matrix factorization framework.

An extension is Bayesian PMF (BPMF) [Salakhutdinov and Mnih, 2008b] where priors are placed on the prior mean and covariance of U_i, V_j. Should we decide to learn W in TGP, interesting parallels arise between our model and BPMF. Observe from the following that learning W can be a proxy for learning the prior mean and covariance of U and V, as is done in the BPMF model:

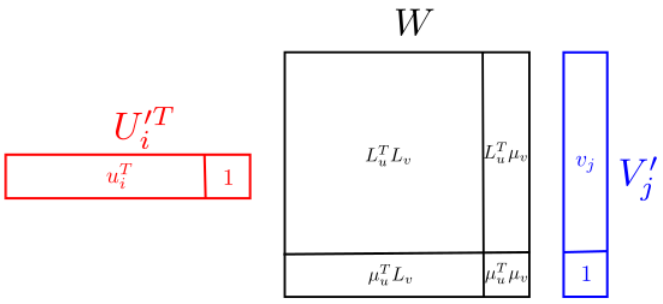


Figure 2.2: Bayesian PMF reparametrised.

$$U_i \sim \mathcal{N}(\mu_u, \Lambda_u), V_j \sim \mathcal{N}(\mu_v, \Lambda_v) \Rightarrow U_i = \mu_u + L_u u_i, V_j = \mu_v + L_v v_j$$
$$\text{where } u_i, v_j \sim \mathcal{N}(0, I), \Lambda_u = L_u L_u^\top, \Lambda_v = L_v L_v^\top$$
$$\Rightarrow U_i^\top V_j = \mu_u^\top \mu_v + \mu_u^\top L_v v_j + u_i^\top L_u^\top \mu_v + u_i^\top L_u^\top L_v v_j = U_i'^\top W V_j',$$

where $U_i'^\top = [u_i^\top, 1]$, $W = [L_u^\top L_v, L_u^\top \mu_v; \mu_u^\top L_v, \mu_u^\top \mu_v]$, $V_j' = [v_j; 1]$, as displayed in Figure 2.2. So a full W with standard iid Gaussian priors on U, V can capture the effects of modelling U, V with non-zero means and full covariances for each row of U, V, as in BPMF.

Returning to the case with side information, suppose it is given in the form of vectors $\omega_1(u_i), \omega_2(v_j)$, and that we expect users/movies with similar ω to show similar preferences/be preferred by similar users. For example we can encode the user age into ω_1 and the movie genre into ω_2 and define $\kappa_1(u_i, u_i') = \omega_1(u_i)^\top \omega_1(u_{i'}), \kappa_2(v_j, v_j') = \omega_2(v_j)^\top \omega_2(v_{j'})$. The feature vector is now $\phi_d(u_i) = [a_d e_i^\top, b_d \omega_d(u_i)^\top]^\top$ for $d = 1, 2$, and we have $f(u_i, v_j) = \phi_1(u_i)^\top UWV^\top \phi_2(v_j)$.

2.5 Related Work and Discussion

Modelling data in the form of matrices and tensors has been studied in the field of multi-way data analysis and relational learning. The key idea here is to factorise the data tensor, with two notable forms of factorisation: PARAFAC [Bro, 1997] and Tucker [Tucker, 1966]. There are a few works in these domains that relate to GPs. InfTucker [Xu et al., 2011] uses the Tucker decomposition directly on the data tensor, and use a non-linear transformation of the parameters $U^{(d)}$ for the regression function, contrary to TGP which is linear in the parameters. DinTucker [Zhe et al., 2016a] tries to scale up InfTucker by splitting up the observed tensor into subarrays. Imaizumi and Hayashi [2016] motivate their model using the Parafac decomposition instead of Tucker, expressing the regression function as a sum of products of local functions. These local functions are each modelled by GPs. However the TGP is motivated from a single GP on the input space. Suzuki et al. [2016] again model the regression function as a sum of product of local functions, which live in the RKHS of some kernel, analogous to the feature maps in TGP. However there is no mention of GPs or how their model relates to low-rank tensor decomposition. Zhao et al. [2013] deals with the classification problem where each input is a tensor, so there is one label per tensor. They define a GP over the space of tensors. It is unclear whether they actually use low rank tensor decomposition. For TGP we deal with regression, work in the setting where each element of the data tensor corresponds to a response, and apply Tucker decomposition to the parameters. Zhe et al. [2016b] define a GP over the parameter space whereas TGP is a multilinear expression in the parameters and feature maps, approximating a GP over the input space. In short, these models make completely different assumptions to TGP, and thus are useful for different CF applications - none use side information (it is unclear how this would be possible given their model assumptions) and do not relate to the Bayesian matrix factorisation literature.

There are closer connections between our model and the Stochastic Relational Model [Yu et al., 2006] in relational learning. It is a special case of our model with $W = I$ and $D = 2$. The key differences lie in the inference: we use features to build on the weight-space view of GPs, whereas Yu et al. [2006] work with GPs in the function-space view. This complicates learning for kernels which cannot be expressed as an inner product of features; the authors resort to Laplace approximation for finding maximum likelihood estimates of parameters. For such kernels we use random feature maps (see Appendix 2.8.4), making learning simple and more computationally efficient.

In the domain of matrix factorisation, Lawrence and Urtasun [2009] use a GP Latent Variable Model (GP-LVM) [Lawrence, 2004]. They learn a latent vector for each movie, and pass it through a zero-mean GP with squared exponential (SE) kernel, with one GP per user. In TGP we use one GP for all users and items. For a CF application, they incorporate side information about movies by taking the product of these kernels with a SE kernel in the movie features. Our model is more flexible in that we can take into account both user and item similarities simultaneously.

From the perspective of GP regression, we analyse the regression function of TGP to understand the regression problems for which it will be effective. Recall that the regression function $f(x)$ in (2.5) can be seen as $W \times_{d=1}^{D} \psi_d(x)$, where $\psi_d(x) = U^{(d)\top}\phi_d(x)$ are lower-dimensional features in $\mathbb{R}^r$ (i.e. the $U^{(d)}$ multiplied by $\phi_d(x)$ in Figure 2.1b). With this new formulation, we have:

$$f(x) = W \times_{d=1}^{D} \phi_d(x) = \sum_{i_1,\ldots,i_D=1}^{r} W_{i_1\ldots i_D} \prod_{d=1}^{D} (\psi_d(x))_{i_d}. \tag{2.7}$$

Hence learning W and $\left(U^{(d)}\right)_{d=1}^{D}$ can be interpreted as learning features ψ_d as well as their weights for the regression function, i.e. learning a linear combination of products of these features. In the case where $\psi_d(x)$ is only a function of the d^{th} dimension of x, each $\prod_{d=1}^{D}(\psi_d(x))_{i_d}$ is separable in the dimensions. Modelling data with sums of separable functions has been studied in Beylkin et al. [2009], and its effectiveness for regression is shown by promising results on various synthetic and real data. Such additive models arise frequently in the context of ensemble learning, such as *boosting* and BART [Chipman et al., 2010], where a linear combination of many weak learners is used to build a single strong learner. We may interpret our model in this framework where $\prod_{d=1}^{D}(\psi_d(x))_{i_d}$ are the weak learners that share parameters, and $W_{i_1\ldots i_D}$ are the corresponding weights.

With this alternative interpretation in mind, we may expect TGP to perform well in cases where the data displays an additive structure, with the additive components arising from a product of features on each dimension. Hence we interpret TGP as a modified GP where the approximation acts as a regulariser towards such simpler functions, which can actually lead to enhanced generalisation performance by controlling overfitting. We thus compare its performance to GPs on spatio-temporal data sets where it is reasonable to expect separability in longitude and latitude, or in time and space.

Based on Section 2.3.2, we also see that our model is particularly well-suited to modelling grid-structured data. The difference between our model and that in Saatçi

[2011] is that we have Kronecker structure in the features ϕ, whereas they exploit Kronecker structure on the data. Moreover, our model can deal with data not on a grid, as well as data on a grid with many missing observations, since observations are not needed for constructing the features.

Going back to CF, recall from Section 2.4 that the low-rank matrix factorisation model has $R_{ij} \approx \sum_k U_{ik} V_{jk}$, a sum of a product of parameters(features) in each dimension. TGP generalises this to modelling a linear combination of products of features, hence we may expect it to perform well for this task.

2.6 Experimental Results

Collaborative Filtering We use the MovieLens 100K data[1], which consists of 100,000 ratings in $\{1, \ldots, 5\}$ from 943 users on 1682 movies. User age, gender and occupation are given, as well as the genre of the movies. We represent this side information with binary vectors for $\omega_1(u_i), \omega_2(v_j)$ and use the formulation in (2.8) in Appendix 2.8.5. We bin the age into five categories, and there are 20 occupations and 18 genres. Thus $\omega_1(u_i) \in \mathbb{R}^{5+2+20}$ has 3 non-zero entries, one for each feature, and $\omega_2(v_j) \in \mathbb{R}^{18}$ can have multiple non-zero entries since each movie can belong to many genres. We report the mean and standard deviation of the test RMSE on the five 80:20 train test splits that come with the data, as it will offer a sensible means of comparison with other algorithms. N is too large for HMC, hence we use SGD to obtain MAP estimates for the parameters, and compare different configurations: learning W/fixing it to be the identity and using/not using side information, along with BPMF initialised by PMF[2]. We use mini-batches of size 100, and set $r = 15$ for all models as it gives best results for PMF and BPMF. We used a grid search and cross-validation for tuning hyperparameters, the recommended method in big N settings where the number of hyperparameters is not too large. SGD was not so sensitive to mini-batch size, and finding the range of hyperparameters was straightforward. See Appendix 2.8.5 for details.

From Table 2.1 it is evident that TGP makes good use of side information, since the RMSE decreases significantly with side information. Learning W instead of fixing it helps predictive performance, but does not perform as well as BPMF. One reason is that our Gaussian prior on W is not equivalent to the Gaussian-Wishart priors on the mean and variance of U_i, V_j in BPMF. Another reason is that we are resorting to a

[1] Obtained from http://grouplens.org/datasets/movielens/100k/
[2] Code obtained from http://www.cs.toronto.edu/~rsalakhu/BPMF.html

Table 2.1: Test RMSE on MovieLens100K.

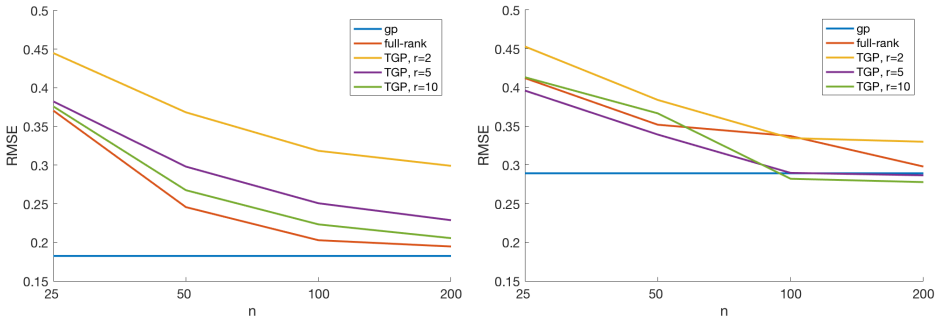
Model	**Test RMSE**
BPMF	0.9024 ± 0.0050
TGP, $W = I$ (PMF)	0.9395 ± 0.0115
TGP, learn W	0.9270 ± 0.0097
TGP, $W = I$, side-info	0.9014 ± 0.0061
TGP, learn W, side-info	$\mathbf{0.8995 \pm 0.0062}$

MAP estimate. If we can instead sample from the posterior and average predictions over these samples, we expect enhanced predictions. However, note that using TGP with side information and learning W, we are able to get comparable/superior results to BPMF, even with a MAP estimate. We expect further improvements not only with sampling but also by using more sophisticated kernels that make better use of the side information; for example, use different hyperparameter coefficients for the different types of features. In so far as comparison was possible, these numbers are comparable to state-of-the-art algorithms in Section 2.5. A direct comparison was not possible as each use different methods for evaluation.

Regression on spatial data We use the California house price data from the 1990 census[3], which consists of average house prices for 20,640 different locations in California. We only use the covariates longitude and latitude, and whiten them along with log-transformed house prices so that each has zero mean and unit variance. We chose this data set as spatial data sometimes exhibit separability in the different dimensions. Moreover the data is clustered in urban areas, hence an additive model with each component describing different sections of California may be desirable. Using a random 50:50 train test split, we report the RMSE of the model on the training set and test set after training. We first fit a GP to the data with a squared expeonential (SE) kernel on each dimension using the GPML toolbox [Rasmussen and Nickisch, 2010], optimising the hyperparameters by maximum likelihood. Then using these hyperparameters we generate RFF for ϕ. See Appendix 2.8.3 and 2.8.4 for details. We implemented both the *full-rank* model and TGP with $n = 25, 50, 100, 200$ on Stan [Stan Development Team, 2012], which uses HMC with the No-U-Turn Sampler (NUTS) [Hoffman and Gelman, 2014] for inference. Note that for both models n refers to the length of features $\phi_d(x)$.

For TGP, we use 300 warmup iterations and a further 300 samples on 4 different chains, and use the mean prediction across the samples. For *full-rank*, we take the

[3]Obtained from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/cadata



(a) Train RMSE (b) Test RMSE

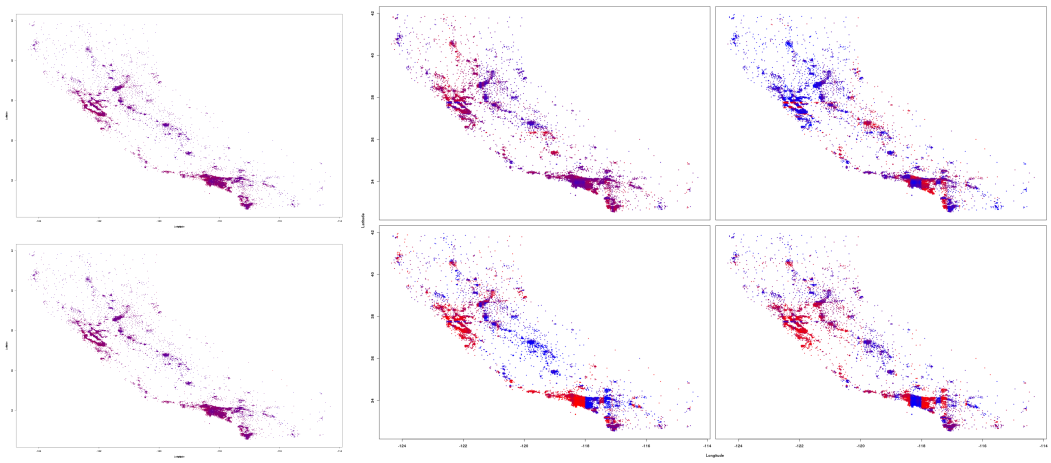
Figure 2.3: RMSE for GP, *full-rank*, and TGP for $r = 2, 5, 10$ for $n = 25, 50, 100, 200$ on the California House Price data

same number of samples and chains, but only use 50 warmup draws as we diagnosed that convergence was reached by this point (looking at the Gelman-Rubin statistic [Gelman and Rubin, 1992] and effective sample size). The convergence statistics for TGP are in Appendix 2.8.6. We can see from Figure 2.3 that some TGP models give lower test RMSE and higher train RMSE than the GP and the *full-rank* model. In fact TGP with $r = 5$ consistently shows higher predictive performance than *full-rank* for all values of n, and for $n \geq 100$ TGP with $r = 10$ outperforms GP. This indicates that TGP is an effective regulariser towards simpler regression functions, namely a linear combination of separable functions. We expect bigger gains for TGP with more warmup iterations, since the convergence diagnostics suggest that TGP hasn't quite fully mixed by 300 iterations.

We further investigate the predictions of TGP by analysing the additive components in the prediction for $r = 2$. We see in Figure 2.4b that the components are quite different. The upper two components show complementary predictions in the Bay area (North-West) and the central area, whereas the bottom two show complementary predictions in the Los Angeles area (South-East). This confirms the hypothesis that the different additive components will learn different sections of the data. See Appendix 2.8.6 for zoomed in plots, and Appendix 2.8.7 for experimental results on spatio-temporal data with grid structure.

2.7 Discussion

We have introduced the Tucker Gaussian Process (TGP), a regression model that regularises a GP towards simpler regression functions, in particular a linear combination of separable functions. We motivate it as a solution to Collaborative Filtering (CF),



(a) True values/TGP Predictions

(b) Additive components for TGP

Figure 2.4: (a) Top: Heatmap of true log house price values. Bottom: TGP predictions for $r = 2, n = 200$. (b) Heatmap showing the four additive components (summands in (2.7)) of predictions for TGP with $r = 2, n = 200$. We only use the last sample in the Markov Chain to get a better indication of the structure. Red indicates high log price and blue indicates low, and the same colour scheme is applied to all four subplots. To accentuate the differences in the predictive values, we colour values by the percentile they belong to instead of a uniform colouring. See Appendix 2.8.6 for the uniform colouring.

by using feature maps and a low-rank Tucker decomposition on the parameters in the weight-space view of GPs. In particular, we have highlighted the effectiveness of TGP in CF with side-information, a domain where outputs can be effectively modelled as a linear combination of functions separable in the covariates. We believe that this is the largest contribution of our paper: after showing that PMF and BPMF are special cases of the TGP, we extended it to exploit the user/item side information for better predictions; the kernel of the GP we approximate can be designed to encode similarities between different users and items, a particularly neat and natural method for modelling similarity. In doing so, we bring together matrix factorisation methods and GP methods in CF, as well as scaling up GP methods in CF. We confirm experimentally that side information enhances the predictive performance of TGP in collaborative filtering.

We have also shown that for problems where one might expect separability in the covariates such as prediction for spatio-temporal data sets, the TGP effectively controls overfitting and outperforms GPs in prediction. We also point out that exact Cholesky features can be used with TGP in the case of grid-structured data, and

random feature maps can be used for arbitrary kernels.

2.8 Appendix

2.8.1 Elementwise convergence of TGP to $\mathcal{N}(0,1)$

Definition 2.8.1. ***Martingale Difference Sequence*** *A martingale difference sequence with respect to a filtration $(\mathcal{F}_p)_{p\in\{0,1,\dots,r\}}$ is a real-valued sequence of random variables $X_1,\dots,X_r$ that satisfies:*

1. *X_p is $\mathcal{F}_p$ measurable*
2. *$\mathbb{E}(|X_p|) < \infty$*
3. *$\mathbb{E}(X_p|\mathcal{F}_{p-1}) = 0$ a.s.*

for all $p \in \{1,\dots,r\}$.

Theorem 1 (**Martingale Central Limit Theorem** [Hall and Heyde, 2014])**.** *Let $X = \{X_1,\dots,X_r\}$ be a sequence of random variables satisfying the following conditions:*

1. *X is a martingale difference sequence with respect to filtration $(\mathcal{F}_p)_{p\in\{0,1,\dots,r\}}$*
2. *$\sum_{p=1}^r \mathbb{E}(X_p^2|\mathcal{F}_{p-1}) \xrightarrow{p} 1$ as $r\to\infty$.*
3. *$\sum_{p=1}^r \mathbb{E}(X_p^2\mathbb{I}(|X_p|>\epsilon)|\mathcal{F}_{p-1}) \xrightarrow{p} 0$ as $r\to\infty$ $\forall\epsilon>0$.*

Then the sums $S_r = \sum_{p=1}^r X_p \xrightarrow{d} \mathcal{N}(0,1)$ as $r\to\infty$.

Proposition 1. *Let n by r matrices $U^{(d)} \overset{iid}{\sim} \mathcal{N}(0,\frac{1}{r}I)$ for $d=1,\dots,D$, and let $W\sim\mathcal{N}(0,I)$ where $W\in\mathbb{R}^{r\times\dots\times r}$ is a D-dimensional tensor. Then each element of $W\times_{d=1}^D U^{(d)\top}$ converges in distribution to $\mathcal{N}(0,1)$ as $r\to\infty$.*

Proof. Suppose first that $D=2$. It suffices to show that

$$u,v\in\mathbb{R}^r, W\in\mathbb{R}^{r\times r}, u,v \overset{iid}{\sim}\mathcal{N}(0,I), W\sim\mathcal{N}(0,I)$$
$$\Rightarrow u^\top W v \xrightarrow{d}\mathcal{N}(0,1) \text{ as } r\to\infty$$

We define for each $r \in \mathbb{N}$:

$$S_0 = 0$$
$$S_p := \sum_{i,j=1}^{p} u_i W_{ij} v_j$$
$$X_p := S_p - S_{p-1} = u_p W_{pp} v_p + \sum_{i=1}^{p-1} u_p W_{pi} v_i + u_i W_{ip} v_p$$
$$\mathcal{F}_0 := \{\emptyset, \Omega\} \text{ where } \Omega \text{ is the sample space for the RVs } u, v, W$$
$$\mathcal{F}_p := \sigma(u_i, v_j, W_{ij})_{i,j=1}^{p}, \text{ the sigma algebra generated by these random variables}$$

for $p \in \{1, \ldots, r\}$.

So we have that $S_r = u^\top W v$, hence it suffices to check conditions 1,2,3 in the Martingale CLT.
We first show 1, that X is a martingale difference sequence. It is clear that X_p is $\mathcal{F}_p$ measurable by definition of $\mathcal{F}_p$. To show that X is integrable, we have:

$$\begin{aligned}
\mathbb{E}(|X_p|) &\leq \mathbb{E}(|u_p W_{pp} v_p|) + \sum_{i=1}^{p-1} \mathbb{E}(|u_p W_{pi} v_i|) + \mathbb{E}(|u_i W_{ip} v_p|) \\
&\leq \sqrt{\mathbb{E}(u_p^2 W_{pp}^2 v_p^2)} + \sum_{i=1}^{p-1} \sqrt{\mathbb{E}(u_p^2 W_{pi}^2 v_i^2)} + \sqrt{\mathbb{E}(u_i^2 W_{ip}^2 v_p^2)} \\
&= \frac{1}{r} + (p-1)\left(\frac{1}{r} + \frac{1}{r}\right) < \infty,
\end{aligned}$$

by the inequality $\mathbb{E}(|X|)^2 \leq \mathbb{E}(X^2)$ (shown using convexity of $g : x \to x^2$ and Jensen's inequality) and independence of u, v, W. Also we have:

$$\begin{aligned}
\mathbb{E}(X_p | \mathcal{F}_{p-1}) &= \mathbb{E}(u_p W_{pp} v_p) + \sum_{i=1}^{p-1} \mathbb{E}(u_p W_{pi}) v_i + u_i \mathbb{E}(W_{ip} v_p) \\
&= 0,
\end{aligned}$$

since u_p, v_p, W_{pi}, W_{ip} are independent of $\mathcal{F}_{p-1}$ and have zero mean. Hence X forms a martingale difference sequence.

To verify the next two conditions, we first prove a lemma that will help us do so. This is the generalisation of Chebyshev's inequality to higher moments:

Lemma 2. *Suppose X is a random variable with bounded n^{th} moment for some $n \in \mathbb{N}$. Then*
$\mathbb{P}(|X - \mathbb{E}(X)| > \epsilon) \leq \frac{\mathbb{E}(|X - \mathbb{E}(X)|^n)}{\epsilon^n} \ \forall \epsilon > 0.$

Proof. Without loss of generality, assume $\mathbb{E}(X) = 0$. Then

$$\mathbb{P}(|X| > \epsilon) = \mathbb{E}[\mathbb{I}(|X| > \epsilon)] = \frac{1}{\epsilon^n}\mathbb{E}[\epsilon^n \mathbb{I}(|X| > \epsilon)] \leq \frac{1}{\epsilon^n}\mathbb{E}[|X|^n \mathbb{I}(|X| > \epsilon)] \leq \frac{\mathbb{E}[|X|^n]}{\epsilon^n}.$$

□

Note Lemma 2 shows that convergence in L^n implies convergence in probability. So to show conditions 2 and 3 of the martingale CLT, it suffices to show that the expectations of the quantities on the left hand sides converge to the right hand side as scalars:

2'. $\sum_{p=1}^{r} \mathbb{E}(X_p^2) \to 1$ as $r \to \infty$.

3'. $\sum_{p=1}^{r} \mathbb{E}(X_p^2 \mathbb{I}(|X_p| > \epsilon)) \to 0$ as $r \to \infty$ $\forall \epsilon > 0$.

Let us show 2'. In $\mathbb{E}(X_p^2)$, note that all cross terms in $\mathbb{E}(X_p^2)$ cancel since all terms have mean 0. So we have:

$$\begin{aligned}
\mathbb{E}(X_p^2) &= \mathbb{E}(u_p^2 W_{pp}^2 v_p^2) + \sum_{i=1}^{p-1} \mathbb{E}(u_p^2 W_{pi}^2 v_i^2) + \mathbb{E}(u_i^2 W_{ip}^2 v_p^2) \\
&= \frac{1}{r^2} + (p-1)\left(\frac{1}{r^2} + \frac{1}{r^2}\right) = \frac{2p-1}{r^2} \\
\Rightarrow \sum_{p=1}^{r} \mathbb{E}(X_p^2) &= \frac{2}{r^2}\sum_{p=1}^{r} p - r \cdot \frac{1}{r^2} = \frac{2}{r^2}\frac{(r+1)r}{2} - \frac{1}{r} = 1.
\end{aligned}$$

To show 3', we first note that for a random variable X,

$$\int_{\delta}^{\infty} \mathbb{I}(X > t)dt = (X - \delta)\mathbb{I}(X > \delta) \text{ for } \delta \in \mathbb{R}$$

Setting $X = X_p^2, \delta = \epsilon^2$ and rearranging we have:

$$\begin{aligned}
X_p^2 \mathbb{I}(|X_p| > \epsilon) = X_p^2 \mathbb{I}(X_p^2 > \epsilon^2) &= \epsilon^2 \mathbb{I}(X_p^2 > \epsilon^2) + \int_{\epsilon^2}^{\infty} \mathbb{I}(X_p^2 > t)dt \\
&= \epsilon^2 \mathbb{I}(|X_p| > \epsilon) + \int_{\epsilon}^{\infty} 2s\mathbb{I}(|X_p| > s)ds \text{ by change of variables } t = s^2 \\
\Rightarrow \mathbb{E}[X_p^2 \mathbb{I}(|X_p| > \epsilon)] &= \epsilon^2 \mathbb{P}(|X_p| > \epsilon) + \int_{\epsilon}^{\infty} 2s\mathbb{P}(|X_p| > s)ds.
\end{aligned}$$

Now we would like to use Lemma 2 to upper bound the right hand side. Note we want to use even n such that $\mathbb{E}[|X|^n] = \mathbb{E}(X^n)$, since we know how to compute $\mathbb{E}(X_p^n)$ but not $\mathbb{E}[|X_p|^n]$. Also note that $\mathbb{P}(|X_p| > s)$ can be bounded by $\frac{\mathbb{E}(X_p^n)}{s^n}$. So we want $n > 2$ for the bound on the integral to become finite. Hence we use $n = 4$, and show that $\mathbb{E}(X_p^4)$ is sufficiently small so that even when we sum over $p = 1, \ldots, r$, we have

that the upper bound tends to 0 as $r \to \infty$. First we compute $\mathbb{E}(X_p^4)$. Note from the multinomial theorem:

$$(x_1 + x_2 + \cdots + x_m)^n = \sum_{k_1+k_2+\cdots+k_m=n} \binom{n}{k_1, k_2, \ldots, k_m} \prod_{1\le t\le m} x_t^{k_t}$$

where

$$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!\, k_2! \cdots k_m!}.$$

Applying this to X_p^4 and taking the expectation, we see that the only cross terms that survive are products of even powers of the terms, namely where two of the k_i are 2 and the rest are 0.
Noting $\binom{n}{2,2} = 6$, and that $\mathbb{E}(X^4) = 3\sigma^4$ for $X \sim N(0, \sigma^2)$ we have:

$$\begin{aligned}
\mathbb{E}[X_p^4] &= \mathbb{E}[u_p^4 W_{pp}^4 v_p^4 + \sum_{i=1}^{p-1} u_p^4 W_{pi}^4 v_i^4 + u_i^4 W_{ip}^4 v_p^4] \\
&\quad + 6\mathbb{E}\left[(u_p^2 W_{pp}^2 v_p^2)\left(\sum_{i=1}^{p-1} u_p^2 W_{pi}^2 v_i^2 + u_i^2 W_{ip}^2 v_p^2\right)\right] \\
&\quad + 6\mathbb{E}\left[\sum_{i\neq j}^{p-1} u_p^2 W_{pi}^2 v_i^2 u_p^2 W_{pj}^2 v_j^2 + u_i^2 W_{ip}^2 v_p^2 u_j^2 W_{jp}^2 v_p^2\right] \\
&\quad + 6\mathbb{E}\left[\sum_{i,j=1}^{p-1} u_p^2 W_{pi}^2 v_i^2 u_j^2 W_{jp}^2 v_p^2\right] \\
&= (2p-1)\cdot\frac{3}{r^2}\cdot 3\cdot\frac{3}{r^2} + 6\cdot 2(p-1)\cdot\frac{3}{r^2}\cdot\frac{1}{r}\cdot\frac{1}{r} \\
&\quad + 6\cdot 2\binom{p-1}{2}\frac{3}{r^2}\cdot\frac{1}{r^2} + 6(p-1)^2\frac{1}{r^2}\cdot\frac{1}{r^2} \\
&= \frac{3}{r^4}(8p^2+8p-7).
\end{aligned}$$

So $\mathbb{P}(|X_p| > \epsilon) \le \frac{3}{\epsilon^4 r^4}(8p^2+8p-7)$. Hence

$$\begin{aligned}
\mathbb{E}[X_p^2 \mathbb{I}(|X_p| > \epsilon)] &\le \frac{3}{\epsilon^2 r^4}(8p^2+8p-7) + \int_\epsilon^\infty 2s\frac{3}{s^4 r^4}(8p^2+8p-7)ds \\
&= \frac{3}{r^4}(8p^2+8p-7)\left(\frac{1}{\epsilon^2} + \int_\epsilon^\infty \frac{2}{s^3}ds\right) \\
&= \frac{3C}{r^4}(8p^2+8p-7),
\end{aligned}$$

where $\int_\epsilon^\infty \frac{2}{s^3}ds = C$. So

$$\sum_{p=1}^{r} \mathbb{E}[X_p^2 \mathbb{I}(|X_p| > \epsilon)] \le \frac{C}{r^4}\sum_{p=1}^{r} 8p^2+8p-7 = O\left(\frac{1}{r}\right) \to 0 \text{ as } r \to \infty$$

since $\sum_{p=1}^{r} 8p^2 + 8p - 7 = O(r^3)$.

So we have shown conditions 1,2',3', hence by martingale CLT we have that

$$S_r = u^\top W v \xrightarrow{d} \mathcal{N}(0,1) \text{ as } r \to \infty.$$

We can prove the claim for $D > 2$ in a similar fashion. □

2.8.2 Feature Hashing

Suppose we have features $\phi(x) \in \mathbb{R}^n$. When n is too large, we may use feature hashing [Weinberger et al., 2009] to reduce the dimensionality of ϕ:

Lemma 3. *Let $h : \{1,\dots,n\} \to \{1,\dots,m\}$ be a hash function for $m \ll n$. i.e. $\mathbb{P}(h(i) = j) = \frac{1}{m}\ \forall j \in \{1,\dots,m\}$. Also let $\xi : \{1,\dots,n\} \to \{\pm 1\}$ be a hash function. Define $\bar{\phi}(x) \in \mathbb{R}^m$ as follows: $\bar{\phi}_j(x) = \sum_{i:h(i)=j} \xi(i)\phi_i(x)$*
Then $\mathbb{E}[\bar{\phi}(x)^\top \bar{\phi}(x')] = \phi(x)^\top \phi(x')$, $Var[\bar{\phi}(x)^\top \bar{\phi}(x')] = O(\frac{1}{m})$.

2.8.3 Random Fourier Features

Theorem 4 (Bochner's Theorem[Rudin, 1964])**.** *A stationary kernel k(d) is positive definite if and only if k(d) is the Fourier transform of a non-negative measure.*

For RFF the kernel can be approximated by the inner product of random features given by samples from its spectral density, in a Monte Carlo approximation, as follows:

$$\begin{aligned} k(x-y) = \int_{\mathbb{R}^D} e^{iv^T(x-y)} d\mathbb{P}(v) \propto \int_{\mathbb{R}^D} p(v) e^{iv^T(x-y)} dv &= \mathbb{E}_{p(v)}[e^{iv^T x}(e^{iv^T y})^*] \\ &= \mathbb{E}_{p(v)}[Re(e^{iv^T x}(e^{iv^T y})^*)] \\ &\approx \frac{1}{n}\sum_{k=1}^{n} Re(e^{iv_k{}^T x}(e^{iv_k{}^T y})^*) \\ &= \mathbb{E}_b[\phi(x)^T \phi(y)], \end{aligned}$$

where $\phi(x) = \sqrt{\frac{2}{n}}(cos(v_1{}^T x + b_1), \dots, cos(v_m{}^T x + b_n))$ with spectral frequencies v_k iid samples from $p(v)$ and b_k iid samples from $U[0, 2\pi]$.
For a one dimensional squared exponential kernel $k(x,y) = \sigma_f^2 \exp\left(-\frac{(x-y)^2}{2l^2}\right)$, the spectral density is $\mathcal{N}(0, l^{-2})$. So we use features $\phi(x) = \sigma_f\sqrt{\frac{2}{n}}(cos(v_1{}^T x + b_1), \dots, cos(v_m{}^T x + b_n))$ where v_k iid samples from $\mathcal{N}(0, l^{-2})$ and b_k iid samples from $U[0, 2\pi]$.

2.8.4 Choice of Feature Map

Cholesky features Consider data with inputs lying on a D-dimensional grid: $x_i \in \mathcal{X} = \times_{d=1}^{D} X^{(d)}$, $|X^{(d)}| = n_d$ finite, where $k_d(x_i, x_j)$ only depends on the values that x_i, x_j take in $X^{(d)}$. The $X^{(d)}$ can be, for example, a finite set of points in Euclidean space, or the set of values a categorical variable can take. Then the Gram matrix K, containing the values of the kernel evaluated at each pair of points on the full grid, can be written as $K = \otimes_{d=1}^{D} K^{(d)}$, a Kronecker product of the Gram matrices $K^{(d)} \in \mathbb{R}^{n_d \times n_d}$ on each dimension [Saatçi, 2011]. The same holds for the Cholesky factor L where $K = LL^\top$: we have $L = \otimes_{d=1}^{D} L^{(d)}$ where $K^{(d)} = L^{(d)} L^{(d)\top} \in \mathbb{R}^{n_d \times n_d}$. Then we define $\phi_d(x_i)$ to be the i^{th} row of $L^{(d)}$, so that $k_d(x_i, x_j) = K_{ij}^{(d)} = \phi_d(x_i)^\top \phi_d(x_j)$. In general a Cholesky decomposition for an m by m matrix takes $O(m^3)$ to compute. Thus $\phi_d(x_i)$ for $i = 1, \ldots, N$ require $O(n_d^3)$ to compute in total. Hence the computation of features become feasible even for large N as long as the n_d are reasonably small.

Random feature maps In most cases the data does not lie on a grid, nor can k_d be expressed as the inner product of finite feature vectors. In this case we can use random feature maps $\phi_d : \mathcal{X} \to \mathbb{R}^n$ where $\mathbb{E}[\phi_d(x)^\top \phi_d(x')] = k_d(x, x')$. An example is random Fourier features (RFF) [Rahimi and Recht, 2007] for stationary kernels, where $\mathbb{V}[\phi_d(x)^\top \phi_d(x')] = O(\frac{1}{n})$. So we are introducing a further approximation $k_d(x, x') \approx \phi_d(x)^\top \phi_d(x')$, with more accurate approximations for larger n. This is feasible even for large N as $\phi_d(x)$ only takes $O(n)$ computation. See Appendix 2.8.3 for details. For non-stationary kernels, we can obtain features by Nyström methods [Williams and Seeger, 2001, Drineas and Mahoney, 2005], which use a set of n inducing points to approximate K. The kernel is evaluated for each pair of inducing points and also between the inducing points and the data, giving matrices K_{nn} and K_{Nn}. Then $\hat{K} \approx K_{Nn} K_{nn}^{-1} K_{Nn}^\top = \Phi^\top \Phi$ where $\Phi = L_{nn}^{-1} K_{Nn}^\top$. Hence the columns of Φ can be defined to be the Nyström features.

2.8.4.1 Learning Algorithms for TGP

We give detailed derivations of various inference algorithms for the TGP. We have a set of N observations $y_i \in \mathbb{R}$ corresponding to a set of inputs $x_i \in \mathcal{X}$, and we wish to regress $y = (y_i)_{i=1}^N$ on $X = (x_i)_{i=1}^N$. We assume that the data generating mechanism takes the form

$$y = f(X) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2 I_N)$$

where $f(X) = (f(x_i))_{i=1}^N \in \mathbb{R}^N$ and also that the regression function takes the following form

$$f(x) = w^\top \otimes_{d=1}^{D} \Big(U^{(d)\top}\phi_d(x)\Big),$$

where

- $W \in \mathbb{R}^{r\times\ldots\times r}$ is a D-dimensional tensor whose entries are iid $\mathcal{N}(0, \sigma_w^2)$
- $w = vec(W)$ is the vector obtained when flattening tensor W, such that $W \times_{d=1}^{D} v_d = w^\top \otimes_{d=1}^{D} v_d \ \forall v_d \in \mathbb{R}^r$
- $(\phi_d(x))_{d=1}^{D}$ are the features in $\mathbb{R}^n$ extracted from x
- $(U^{(d)})_{d=1}^{D}$ are a set of real $n \times r$ matrices with $U_{jl}^{(d)} \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2)$.

We assume $n > r$, and wish to learn w and the $U^{(d)}$ from the data.

Note from the second point that $\nabla_w\Big(W \times_{d=1}^{D} v_d\Big) = \otimes_{d=1}^{D} v_d$. For D=2 for example, if $g(U) = s^\top U t$ for some matrix U and vectors s, t, then $\nabla_u g(U) = s \otimes t$ where $u = vec(U)$.

First we give the complexity for calculating $f(x)$. Computing $\psi_d(x_i) = U^{(d)\top}\phi_d(x_i)$ $\forall d$ requires $O(nrD)$ time, then $w^\top \otimes_{d=1}^{D} \psi_d(x_i)$ takes $O(r^D)$ time. So time for a prediction given ϕ, U, w takes $O(nrD + r^D)$.

The quantity of interest for MAP and HMC is the log joint distribution $p(y, U, w) = p(y|U, w)p(U)p(w)$. In full this is:

$$\log p(y|U,w) + \log p(U) + \log p(w) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2 - \frac{1}{2\sigma_u^2}\sum_{k=1}^{D} tr(U^{(k)T}U^{(k)}) - \frac{1}{2\sigma_w^2} w^\top w$$

This has the following derivatives:

$$\begin{aligned}
\nabla_w \log p(w) &= -w \\
\nabla_{U^{(k)}} \log p(U) &= -rU^{(k)} \\
\nabla_w \log p(y_i|U,w) &= \frac{1}{\sigma^2}(y_i - f(x_i)) \otimes_{d=1}^{D} \psi_d(x_i) \\
\nabla_{u^{(k)}} \log p(y_i|U,w) &= \frac{1}{\sigma^2}(y_i - f(x_i))\phi_k(x_i) \otimes \Big(W \times_{d\neq k} \psi_d(x_i)\Big),
\end{aligned}$$

with the following definitions:

- $u^{(k)} = vec(U^{(k)}) \in \mathbb{R}^{nr}$
- $(W \times_{d\neq k} v_d)_l := W \times_{d=1}^{D} v'_d$ where $v'_d = v_d$ for $d \neq l$ and $v'_l = e_l \in \mathbb{R}^r$, the unit vector with non-zero at the l^{th} entry.

The last derivative holds since $f(x) = \phi_k(x)^\top U^{(k)}(W \times_{d\neq k} \psi_d(x_i))$.

Computing $W \times_{d\neq k} \psi_d(x_i)$ takes $O(r^D)$ for each k, hence $O(r^D D)\ \forall k$. So we have that using mini-batches $\{x_{t1}, \ldots, x_{tm}\}$ for SGD, we have the following updates for MAP:

$$w \leftarrow w + \frac{\epsilon^w}{2}\left(\nabla_w \log p(w_t) + \frac{N}{m}\sum_{i=1}^{m} \nabla_w \log p(y_{ti}|x_{ti}, w, U)\right)$$
$$u^{(k)} \leftarrow u^{(k)} + \frac{\epsilon^k}{2}\left(\nabla_{u^{(k)}} \log p(U) + \frac{N}{m}\sum_{i=1}^{m} \nabla_{u^{(k)}} \log p(y_{ti}|x_{ti}, w, U)\right),$$

with time complexity $O(m(nrD + r^D D))$.

Gathering the parameters into a vector $\theta = (w, U^{(1)}, \ldots, U^{(k)})$, the HMC algorithm runs as follows:

1. Initialise MC by drawing $\theta_0 = (w_0, U_0^{(1)}, ..., U_0^{(D)})$ from its prior distribution.

2. For $t = 0, ..., T$:

   (a) Initialise $p \sim \mathcal{N}(0, I_Q)$, $V^{(k)} \sim \mathcal{N}(0, I_{n\times r}) \quad \forall k$,
   $H_t \leftarrow -\log p(w_t) - \sum_{i=1}^{N} \log p(y_i|x_i, \theta_t) + \frac{1}{2}\sum_{k=1}^{D} tr(V^{(k)T}V^{(k)}) + \frac{1}{2}p^T p$
   $\theta = (w, U^{(1)}, ...U^{(D)}) \leftarrow \theta_t = (w_t, U_t^{(1)}, ...U_t^{(D)})$

   (b) For $l = 1, ..., L$:

      i. $p \leftarrow p + \frac{\epsilon_t^w}{2}\left(\nabla_w \log p(w_t) + \sum_{i=1}^{N} \nabla_w \log p(y_i|x_i, \theta_t)\right)$
      For $k = 1, ..., D$:
      $V^{(k)} \leftarrow V^{(k)} + \frac{\epsilon_t^k}{2}\left(\sum_{i=1}^{N} \nabla_{U^{(k)}} \log p(y_i|x_i, \theta)\right)$

      ii. $w \leftarrow w + \epsilon_t^w p$
      For $k = 1, ..., D$:
      $u^{(k)} \leftarrow u^{(k)} + \epsilon_t^w V^{(k)}$

      iii. same as i.

   (c) $H^* \leftarrow -\log p(w) - \sum_{i=1}^{N} \log p(y_i|x_i, \theta) + \frac{1}{2}\sum_{k=1}^{D} tr(V^{(k)T}V^{(k)}) + \frac{1}{2}p^T p$
   $u \sim Unif[0, 1]$
   If $u \leq \exp(H_t - H^*)$
   $\theta_{t+1} = (w_{t+1}, U_{t+1}^{(1)}, ..., U_{t+1}^{(D)}) \leftarrow \theta = (w, U^{(1)}, ..., U^{(D)})$
   else
   $\theta_{t+1} \leftarrow \theta_t$.

From previous computations, it is easy to see that each update requires $O(LN(nrD + r^D D))$ operations.

2.8.5 Collaborative Filtering

2.8.5.1 Using Binary Vectors for Side Information

Note if the side information $\omega_1(u_i), \omega_2(v_j)$ are binary vectors with non-zeros at indices $\mathcal{I}_i, \mathcal{J}_j$ respectively, we have:

$$f(u_i, v_j) = (a_1 U_i + b_1 \sum_{k \in \mathcal{I}_i} U_{n_1+k})^T W (a_2 V_j + b_2 \sum_{k \in \mathcal{J}_j} V_{n_2+k})$$

which can be reparametrised to:

$$f(u_i, v_j) = a(U_i + b \sum_{k \in \mathcal{I}_i} U_{n_1+k})^T W (V_j + c \sum_{k \in \mathcal{J}_j} V_{n_2+k}). \tag{2.8}$$

2.8.5.2 Hyperparameter tuning for MovieLens 100K

Hyperparameters were tuned on the following values. For PMF and fixed W TGP: $\sigma_u = [0.3, 0.1, 0.03], \sigma^2 = [1.0, 0.1, 0.01, 0.001], \epsilon_u = [10^{-5}, 10^{-6}, 10^{-7}]$ where ϵ_u, ϵ_w are the step sizes for SGD on U/V and W respectively. We noticed that for a fixed W the model overfits quickly in less than 30 epochs, whereas when learning W the test RMSE decreases steadily. So we used a different grid of parameters for tuning the models where W is learned: $\sigma_u = [0.3, 0.1], \sigma^2 = [1.0, 0.75], \epsilon_u, \epsilon_w = [10^{-5}, 10^{-6}]$. For models with side information, we tuned on $a = [0.25, 0.5, 0.75], b, c = [0.15, 0.3, 0.45]$.

2.8.6 California House Prices Data

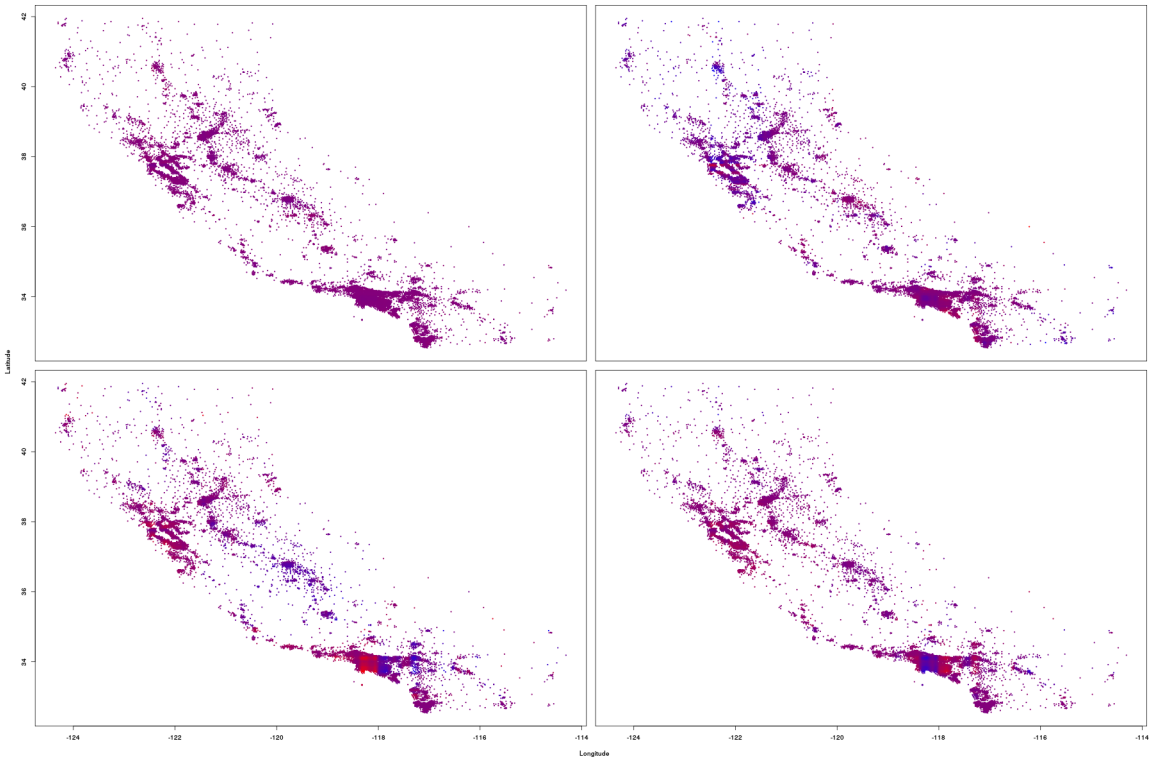


Figure 2.5: Heatmap showing the four additive components of predictions of the last sample of TGP for $r = 2, n = 200$, using uniform colouring scheme.

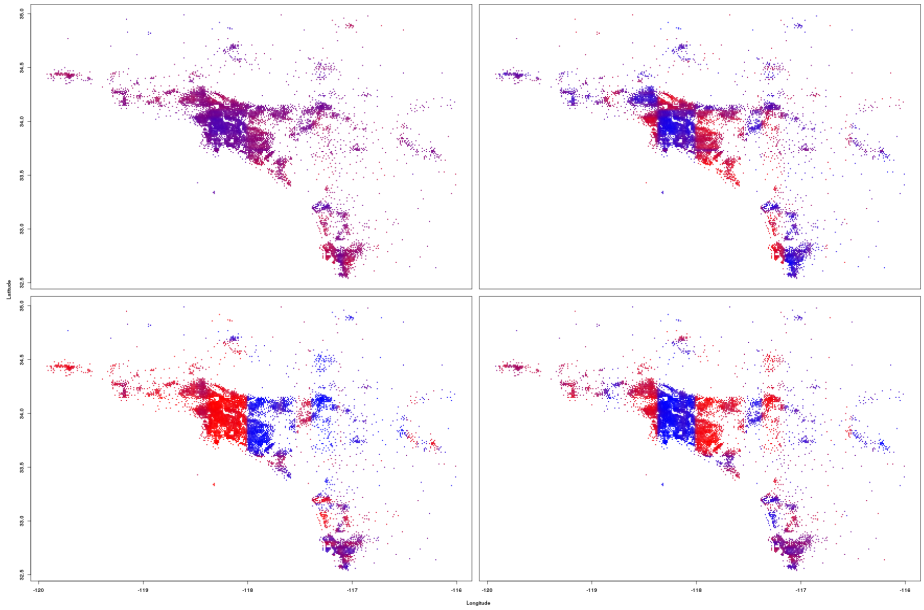


Figure 2.6: Zoom in on LA area of Figure 2.4b.

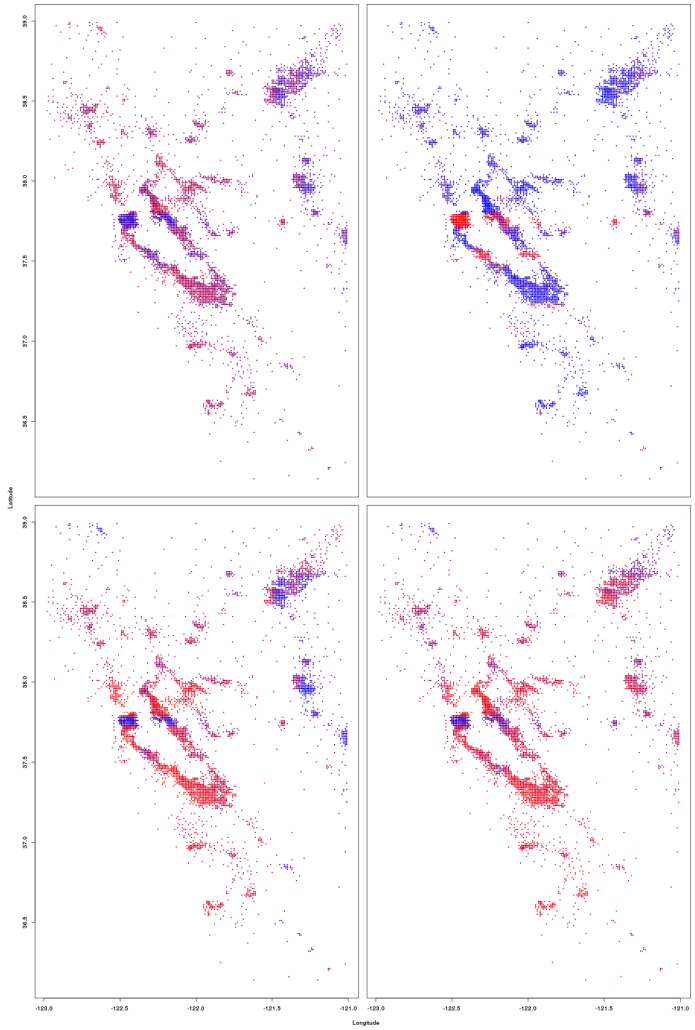


Figure 2.7: Zoom in on Bay area of Figure 2.4b.

Table 2.2: Mean and standard deviation of Gelman Rubin statistic for HMC on TGP.

Model	$n = 25$	$n = 50$	$n = 100$	$n = 200$
TGP, $r = 2$	2.67 ± 1.34	2.55 ± 1.37	2.10 ± 0.70	1.92 ± 0.67
TGP, $r = 5$	1.06 ± 0.27	1.06 ± 0.19	1.15 ± 0.11	1.11 ± 0.11
TGP, $r = 10$	1.00 ± 0.03	1.02 ± 0.04	1.01 ± 0.02	1.06 ± 0.03

Table 2.3: Mean and standard deviation of Effective Sample Size (out of 1200) for HMC on TGP.

Model	$n = 25$	$n = 50$	$n = 100$	$n = 200$
TGP, $r = 2$	230 ± 459	5 ± 17	11 ± 81	12 ± 74
TGP, $r = 5$	244 ± 118	121 ± 70	34 ± 64	42 ± 79
TGP, $r = 10$	692 ± 152	196 ± 93	310 ± 111	96 ± 165

2.8.7 Irish Wind Data

Regression on spatio-temporal data with grid structure We use the Irish wind data [4] giving daily average wind speeds for 12 locations in Ireland between 1961 and 1978 (78,888 observations). We only use the covariates longitude, latitude and time. Note a 2D grid structure arises for the data when we treat the spatial covariates as one dimension and time as another. Again we whiten each covariate and observations, and use 20,000 randomly chosen data points for training and the rest for test. Using an isotropic SE kernel for space, and the sum of a periodic kernel and a SE kernel for time (to model annual periodicity and global trend), we first fit a GP efficiently exploiting the grid structure [Saatçi, 2011]. The optimised hyperparameters are then used to construct Cholesky features. Again we use NUTS for inference on both the *full-rank* model and TGP, using 4 chains with 100 warmup draws and 100 samples.

Table 2.4: Train/Test RMSE on Irish wind data.

Model	**Train RMSE**	**Test RMSE**
GP	4.8822	4.9915
Full-rank	4.8816	4.9898
TGP, $r = 2$	4.9120	4.9753
TGP, $r = 5$	4.8996	**4.9735**
TGP, $r = 10$	4.8913	4.9754

All models show good convergence after 100 warmup draws, indicated by the aforementioned convergence diagnostics. Looking at Table 2.4, we see similar patterns in the results for the wind data as for the house prices data: the GP, which is equivalent to the *full-rank* model with Cholesky features (confirmed by similar train/test RMSE), shows lower training error than TGP, whereas TGP shows superior predictive performance. These results again suggest that TGP is an effective regulariser towards simpler regression functions compared to GPs.

[4] Obtained from http://www.inside-r.org/packages/cran/gstat/docs/wind

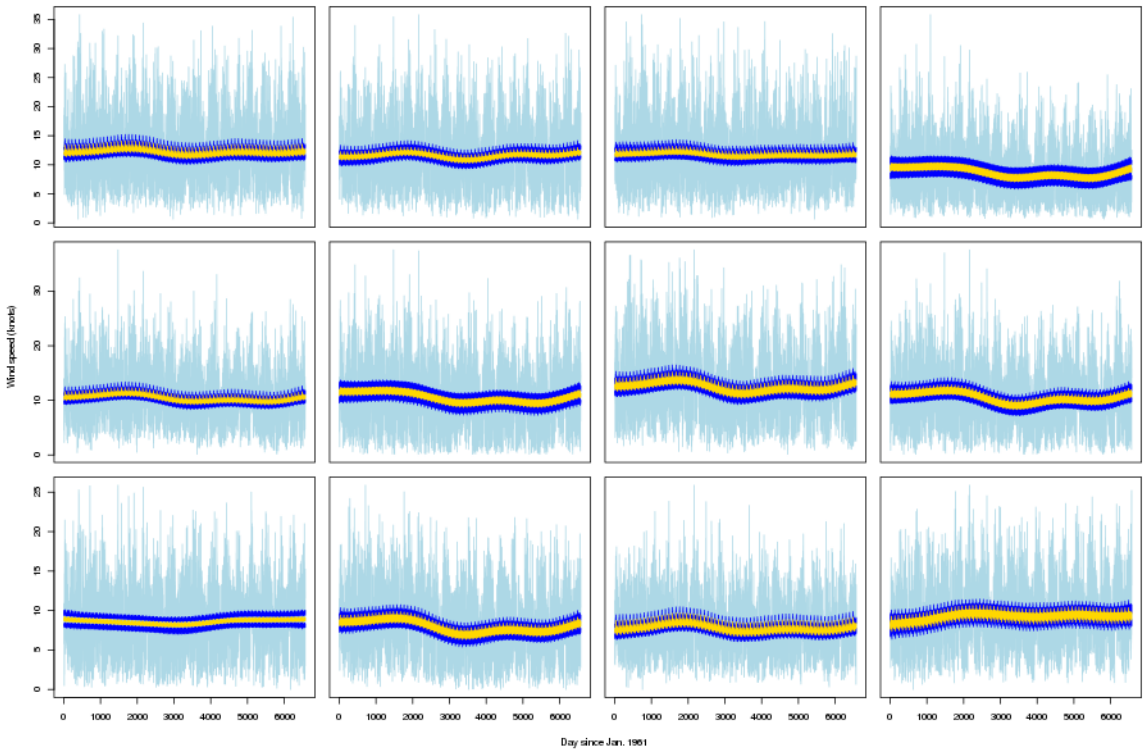


Figure 2.8: The predictions for TGP with $r = 5$ on the 12 locations. The light blue lines are the true observations, the yellow are the mean predictions, and the blue show 2.5% and 97.5% percentiles of predictions for samples.

2.8.8 Future Work

Note that TGP can easily be extended to non-Gaussian likelihoods, since all we need for SGD and HMC is the likelihood and priors to be analytic and differentiable in the parameters. For very high dimensions where even the r^D entries in W are undesirable, we can use a sparse representation of W with say Q non-zeros. All derivations carry forward, and we obtain time complexity $O(m(nrD + QD))$ for gradient computations in SGD. It would be interesting to compare TGP against other algorithms suitable for high-dimensional data. Furthermore, it would be desirable to have a sampling algorithm that scales sub-linearly, to benefit from the Bayesian approach to learning when N is large and HMC is infeasible. One example is Stochastic Gradient Langevin Dynamics (SGLD) [Welling and Teh, 2011b] among many other Stochastic Gradient MCMC [Ma et al., 2015] algorithms. We have also tried mean-field variational inference, but results were poor compared to HMC. Moreover a more efficient method of tuning hyperparameters than by cross-validation would be ideal, especially for big N settings with many kernel hyperparameters. One potential solution is the fully Bayesian

approach, learning hyperparameters directly by imposing priors and sampling or MAP. We leave these extensions for future work.

Chapter 3

Structured Variationally Auto-encoded Optimization

This chapter is based on the paper

- **Xiaoyu Lu**, Javier González, Zhenwen Dai and Neil Lawrence
  Structured Variationally Auto-encoded Optimization
  http://proceedings.mlr.press/v80/lu18c.html
  International Conference on Machine Learning (ICML) 2018.

in which I am the main contributor.

3.1 Abstract

We tackle the problem of optimizing a black-box objective function defined over a highly-structured input space. This problem is ubiquitous in machine learning. Inferring the structure of a neural network or the Automatic Statistician (AS), where the kernel combination for a Gaussian process is optimized, are two of many possible examples. We use the AS as a case study to describe our approach, that can be easily generalized to other domains. We propose an *Structure Generating Variational Auto-encoder* (SG-VAE) to embed the original space of kernel combinations into some low-dimensional continuous manifold where Bayesian optimization (BO) ideas are used. This is possible when structural knowledge of the problem is available, which can be given via a simulator or any other form of generating potentially good solutions. The right *exploration-exploitation* balance is imposed by propagating into the search the uncertainty of the latent space of the SG-VAE, that is computed using variational inference. The key aspect of our approach is that the SG-VAE can be used to bias the search towards relevant regions, making it suitable for transfer learning tasks.

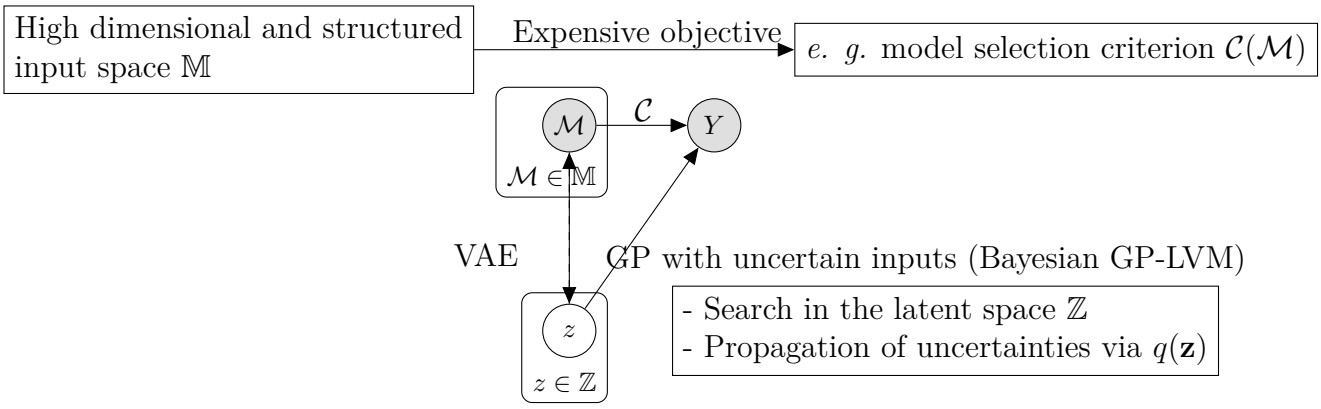


Figure 3.1: *Main elements of* SVO: An expensive objective function needs to be optimized in a in structured input space. A Variational auto-encoder is trained to learn a latent space $\mathbb{Z}$ using data produced by a context-free grammar. BO is applied over the latent space via a GP model with uncertain inputs to find the optimal kernel combination. The uncertainty of the latent space, computed using variational inference techniques, is used in the search to balance exploration and exploitation.

Several experiments in various application domains are used to illustrate the utility and generality of the approach described in this work.

3.2 Introduction

In science and engineering, optimization problems with very structured input domains are ubiquitous. In machine learning we have a few examples. For instance, in deep learning, the architecture of neural networks is defined by a large number of parameters, like the number of layers, activation functions, etc. These parameters show conditional dependencies, which makes the problem specially hard to treat as different network configurations cannot be represented in a unique Euclidean space [Jenatton et al., 2017, Swersky et al., 2013]. In kernel-based methods, the Automatic statistician (AS) is another example. With a Gaussian process (GP) as the class of models of choice, the goal is to automatically select the best kernel combination to explain a data set, which is chosen by enumerating a countably infinite set of arbitrarily complex kernels composed via additions and multiplication of simple ones [Duvenaud et al., 2013]. Interestingly, the final goal of the AS is not just to fit a model to data, but to write a report that uses the type of combination as the main element to interpret the data. Various model selection criteria like the BIC, the model evidence or the mean squared error have been used in this domain. The complexity of these measures is $\mathcal{O}(n^3)$ for n the number of sample points, which makes the problem unsolvable by brute force.

These examples, among others that we will discuss later, can be formalized as optimization problems in which an expensive black-box objective function, typically assumed 'well behaved', is optimized in a highly structured input space. By 'highly structured' we mean that the solutions cannot be trivially represented in a Euclidean input space, making standard optimization approaches impractical. Instead, they belong to a tree or to other non-Euclidean domains. In this paper we are interested in these problems. More specifically, we study how their structure can be leveraged to address them successfully. To simplify the description of our methodology, we will use the AS a case-study, although the methods and ideas proposed here can easily be extended to other domains. To show this, in the experimental section of this work we deal with a problem where the goal is to find the best textual description of a Minecraft image produced by text generator.

Before we get into further details, we start by formalizing the case study, the AS problem, around which we will provide the details of our approach.

3.2.1 Case Study: the Automatic Statistician

In AS the goal is to solve a supervised learning problem given a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^p$ are the inputs and $y_i \in \mathbb{R}$ are the outputs. Let $\mathbb{M}$ be the class of models that we will use to explain the data. In particular, we consider $\mathbb{M}$ to be the family of GPs with different kernel combinations. We denote by $\mathcal{M}$ each model of this set and we denote by $\mathcal{B}_\mathcal{M}$ its associated parameter space. The models in $\mathbb{M}$ represent different structural assumptions about the data such as trends or periodicity. The values of $\beta \in \mathcal{B}$, the model hyper-parameters, differentiate models within the same family.

The goal of AS is to select one single model from $\mathbb{M}$ that explains the data $\mathcal{D}$ the best. We denote by $\mathcal{C} : \mathbb{M} \to \mathbb{R}$ some 'goodness of fit measure' that quantifies the quality of the fit. We keep $\mathcal{C}(\mathcal{M})$ generic for the moment, and we just assume that is expensive enough such that it is only feasible to evaluate in a few models in $\mathbb{M}$. The problem reduces to finding

$$\mathcal{M}_\beta^{opt} := \arg\max_{\mathcal{M}\in\mathbb{M}} \mathcal{C}(\mathcal{M}). \tag{3.1}$$

Problem (3.1) is not suitable to be solved using Bayesian optimization (BO) [Shahriari et al., 2016]. The reason is that the input space of Problem (3.1) is very structured, highly dimensional, and non-continuous so most BO methods will break. The key idea of the approach presented in this paper is to transform Problem (3.1) into a problem that can be handled with standard BO methods.

3.2.2 Related Work

The idea of using BO in high dimensional and structured spaces has already been explored in the literature. Random projections [Wang et al., 2016b], other generative models like deep Gaussian processes [Dai et al., 2016] and combinations of optimization and sampling strategies [Abbati et al., 2017] have been developed recently. Bayesian optimization methods able to deal with hierarchical dependencies have also been proposed [Jenatton et al., 2017, Swersky et al., 2013]. Some interesting applications are the design of molecules [Kusner et al., 2017] and genes [Gonzalez et al., 2014]. Kusner et al. [2017] is closely related to our work: they also use a context-free grammar and a variational auto-encoder to learn a low dimensional representation of the molecules, followed by Bayesian optimisation in that space. But we differ from them in a number of ways: 1) we propagate uncertainties from the variational auto-encoder into the search of Bayesian optimisation; 2) we hard coded the likelihood such as any output from the decoder is a valid kernel representation.

Regarding AS, a few approaches have followed the original compositional approach [Grosse et al., 2012, Duvenaud et al., 2013]. Kim and Teh [2016] scales this method to big data scenarios by using sparse GPs. Hwang and Choi [2015] developed relational kernel learning methods. Malkomes et al. [2016] proposed, to the best of our knowledge, the first approach that uses BO in this context. A parametric measure of kernel similarity, the *Hellinger distance*, is used to guide the search over a selected sets of kernel combinations.

3.2.3 Contributions and Outline

We use the idea that kernel combinations can be expressed as operations of a context-free grammar [Hopcroft et al., 2006]. This is used to simulate combinations with certain properties or structure, similarly to the work of Kusner et al. [2017] where a grammar is used to generate molecules. For instance, if we know a kernel that is likely to appear in the final combination (if a time series has periodicity, for instance), we can leverage the times it appears in the simulations. The idea is to produce data that represent well the set of feasible solutions of the problem and use it to learn a low-dimensional manifold in which the search for the optimal solution takes place. To this end, we learn a latent variable model. We used a Variational Auto-encoder (VAE) [Kingma and Welling, 2013] for practicality, although any other continuous latent variable model could be considered. With the problem mapped into a low-dimensional space, BO can be used to find the best combination, circumventing the issues described

Algorithm 1 Context-free grammar for kernel expressions generation.

Input: N_{max}, $p_{\mathcal{B}}$, $\mathcal{B}$, $p_{\mathcal{O}}$, $\mathcal{O}$, $\mathcal{S} = \emptyset$.
for k $< N_{max}$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} + B$ for $B \in \mathcal{B}$, select kernel with probability $p_{\mathcal{B}}$.
  $\mathcal{S} \leftarrow \mathcal{S} + O$ for $O \in \mathcal{O}$, select operation with probability $p_{\mathcal{O}}$.
  Break when O is *Stop*.
end for

in Section 3.2.1. As there exists uncertainty associated to the learned latent space, we will show how variational inference can be used to map it into the search, which has a positive effect balancing exploration and exploitation. See Figure 3.1 for a graphical description of the main elements of the approach described in this work. The main contributions are:

- A new Variational Auto-encoder, called 'Structure Generating Variational Auto-encoder' (SG-VAE). We describe and use it in the context of mapping kernel combinations produced by a context-free grammar into a continuous and low-dimensional latent space. Although here we use it in the AS context, it is broadly applicable.
- A variational approximation of the distribution of the latent space of the SG-VAE. This distribution is used to propagate the uncertainty of the SG-VAE into the BO search. A GP with uncertain inputs is used to make this step practical.
- A series of experiments that illustrate the utility of this work in model selection and in a problem where the goal is to find the best textual description of a Minecraft image.

Section 3.3 describes the SG-VAE model and in Section 3.3.4 we detail how the SG-VAE can be used in optimization problems. In Section 5.6, we illustrate its performance with a series of experimental results. In Section 3.5 we include some conclusions and further lines of research derived from this work.

3.3 Variational Auto-encoders for Structured Space Representation

In this section we present a new VAE to map structured input spaces into low-dimensional latent manifold. We describe it in the context of the AS, so the goal is to

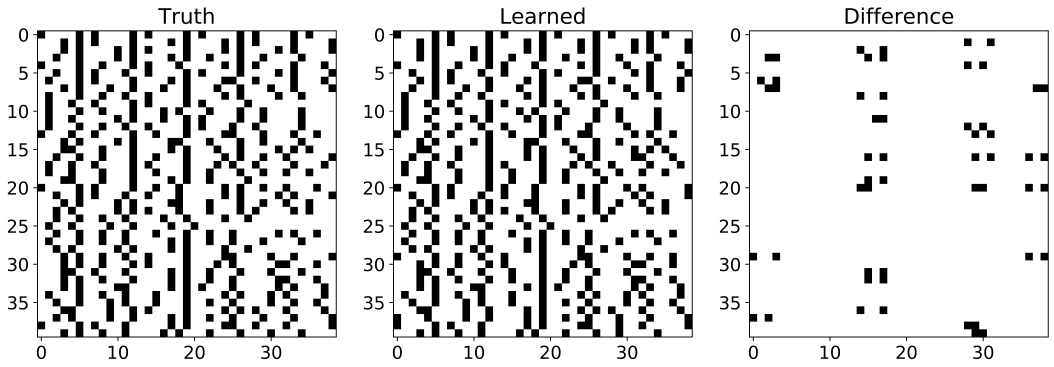


Figure 3.2: Recovery of various kernel combinations after encoding and later decoding them with a SG-VAE for (see experimental section for further details). Black dots represent ones and white dots zeros. *Left*: original one-hot encoding vectors (one per row) representing kernel combinations. *Centre*: vectors produced by the SG-VAE after encoding and decoding (using the mode). *Left*: difference between the original and the mapped vectors.

use it to find good representations of kernel combinations produced by a context-free grammar. We describe the encoder, decoder and a variational approximation of the distribution of the latent variables.

3.3.1 Grammar-based Kernel Representation

It is possible to generate a countably infinite kernel space through closure of kernels via a context-free grammar. Given a set of base kernels $\mathcal{B}$ we can generate an expression (kernel combination) $\mathcal{S}$ by subsequently adding kernels and operations $\mathcal{O}$ (additions, multiplications, replacements or Stop) [Duvenaud et al., 2013, Grosse et al., 2012]. Both the kernels and the operations are chosen according to pre-specified probabilities $p_{\mathcal{B}}$ and $p_{\mathcal{O}}$. See Algorithm 1.

We use 1-hot encoding vectors for both kernels and operations to represent each expression $\mathcal{S}$. Suppose that $\mathcal{B} = \{K_1, K_2, K_3, K_4\}$ is the set of four base kernels and $\mathcal{O} = \{+, \times, \mathit{Stop}\}$ is the set of operations. Any expression $\mathcal{S}$ is transformed into a binary vector by recurrently attaching the 1-hot vectors of each kernel and operation. When the operation is *Stop* the vector is completed with zeros. For instance, in the following example, four kernels are combined using a number of N_{max} operations. Before termination we have:

$$\underbrace{K_2}_{0100}\ \underbrace{+}_{100}\ \underbrace{K_1}_{1000}\ \underbrace{*}_{010}\ \underbrace{K_3}_{0010}\ \underbrace{*}_{010}\ \underbrace{K_1}_{1000}\ \underbrace{\mathit{Stop}}_{001}\ \underbrace{\ldots}_{\mathit{Add}\ 000}$$

This *grammar-based* representation, that we denote by $\mathbf{x}_g$, captures the complexity of the combination [1].

The grammar-based representation accounts for kernel complexity but it does not take into account the differences in the combinations due to the dataset $\mathcal{D}$. For this, a *data-based* representation is proposed in the next section.

3.3.2 Data-based Kernel Representation

We use the vector of distances between the kernel matrices of the base kernels evaluated at the data and the kernel matrices of the combinations. Details about how to deal with the hyper-parameters of the kernels are in the experimental section. As a measure of distance, the *Hellinger distance* could be used [Malkomes et al., 2016]. However its $\mathcal{O}(n^4)$ complexity makes it prohibitively slow. We found that the Frobenius distance works well and it is also very quick to compute. We denote this data-based representation by $\mathbf{x}_d$ which we normalise before the VAE. To be explicit, $\mathbf{x}_d$ is a 4-dimensional vector and its d^{th} entry is equal to $||M - M_{(d)}||$, where M is the gram matrix whose kernel representation is $\mathbf{x}_g$ and $M_{(d)}$ is the gram matrix of the base kernel K_d. Both matrices are evaluated at a subset of the given data. The global representation for each combination is therefore $\mathbf{x} = [\mathbf{x}_g, \mathbf{x}_d]$, which has dimension $(N_{max} + 1)|\mathcal{B}| + (N_{max} - 1)|\mathcal{O}|$ for N_{max} the maximum number of allowed operations (added kernels).

3.3.3 Structure Generating Variational Auto-encoder (SG-VAE)

This section describes a bespoke VAE, the Structure Generating Variational Auto-encoder (SG-VAE), for learning low dimensional representations of the kernel combinations represented by $\mathbf{x} = [\mathbf{x}_g, \mathbf{x}_d]$. The original VAE proposed by [Kingma and Welling, 2013] interprets points $\mathbf{z}$ as elements in a latent space of probabilistic generative model with two main components. A *decoder* given by a likelihood parametrized by θ, $p_\theta(\mathbf{x}|\mathbf{z})$ and a probabilistic *encoder* $q_\phi(\mathbf{z}|\mathbf{x})$ which is a variational approximation to the posterior of $p(z|x)$. The parameters of $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ are optimized jointly by maximizing the the evidence lower bound (ELBO)

$$\mathcal{L}(\phi, \theta; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right],$$

[1] Note that $\mathbf{x}$ here and in the definition of $\mathcal{D}$ represents different vectors.

Algorithm 2 Structured Variational auto-encoded optimization, the SVO algorithm.

Input: Dataset $\mathcal{D}$, precomputed SG-VAE, model selection criterion $\mathcal{C}(\mathcal{M})$ and number of iterations N_{iter}.
1. Select initial point $\mathbf{z}_1$. Compute $q_\gamma(\mathbf{z}_1)$, the mode of the decoder $\tilde{\mathbf{x}}|\mathbf{z}_1$ and $\mathcal{C}(\mathcal{M}_{\tilde{\mathbf{x}}|\mathbf{z}_1})$.
2. Update the dataset $\tilde{\mathcal{D}}_1 = \{(q_\gamma(\mathbf{z}_1), \mathcal{C}(\mathcal{M}_{\tilde{\mathbf{x}}|\mathbf{z}_1}))\}$.
for $j = 1$ **to** N_{iter} **do**
3.1 Fit a GP with uncertain inputs to $\tilde{\mathcal{D}}_j$.
3.2 Maximize the Expected Improvement to obtain $\mathbf{z}_{j+1}$.
3.3 Evaluate the objective at the mode of the decoder $\mathcal{C}(\mathcal{M}_{\tilde{\mathbf{x}}|\mathbf{z}_{j+1}})$.
3.4 Compute the variational approximation of the latent space $q_\gamma(\mathbf{z}_{j+1}|\tilde{\mathbf{x}})$.
3.5 Augment data set: $\tilde{\mathcal{D}}_{j+1} = \{\tilde{\mathcal{D}}_j \cup (q_\gamma(\mathbf{z}_{j+1}), \mathcal{C}(\mathcal{M}_{\tilde{\mathbf{x}}|\mathbf{z}_{j+1}}))\}$.
end for
Returns: Report best obtained latent point $\mathbf{z}^*$ and the associated model $\mathcal{M}_{\mathbf{x}|\mathbf{z}^\star}$.

which is done by using gradient descent as long as $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are differentiable. SG-VAE extends VAE by proposing a type of encoder and decoder that are specialized in representing the structured search space.

3.3.3.1 SG-VAE Encoder

The *encoder* of the SG-VAE is a Gaussian distribution where the mean is the output of a *Multilayer Perceptron (MLP)* . In particular we define

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z} : \mu_\phi, \sigma^2_{en}\mathbf{I})$$
$$\mu_\phi = MLP(x; \phi),$$

where *MLP* represents a feed-forward network with two hidden layers and *tanh* activation function, parametrized by weights ϕ, with σ^2_{en} denotes the variance of the encoder Gaussian distribution.

Therefore, the parameters in the encoder are $\phi = \{\sigma^2, \mathbf{W}_1, \mathbf{W}_2, b_1, b_2\}$. This model can be extended with more intermediate layers and other activation functions. In our experiment we use this configuration unless specified otherwise.

3.3.3.2 SG-VAE Decoder

Although the encoder is somehow standard, the *decoder* is specific to the model representation problem that we are addressing. The challenge is to learn a decoder that always provides a valid kernel representation. To this end, we encode this feature in the model likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ as we next detail. First of all, let

$$\boldsymbol{l} = \exp(\mathbf{W}_4 \text{Relu}(\mathbf{W}_3\mathbf{z} + b_4) + b_5)$$

be the mapping of the decoder given $\mathbf{z}$ in the latent space with parameters $\theta = \{\mathbf{W}_3, \mathbf{W}_4, b_4, b_5\}$. The values of θ need to be obtimized jointly with ϕ.

The likelihood factorizes into two independent components

$$p_\theta(\mathbf{x}|\mathbf{z}) = p_\theta(\mathbf{x}_d|\mathbf{z})p_\theta(\mathbf{x}_g|\mathbf{z})$$

that corresponds to the grammar and data representation $p_\theta(\mathbf{x}_d|\mathbf{z})$ and $p_\theta(\mathbf{x}_g|\mathbf{z})$ respectively. The data representation is modeled with a Gaussian likelihood

$$p_\theta(\mathbf{x}_d|\mathbf{z}) \sim \mathcal{N}(\mathbf{x}_d : \mu_\theta, \sigma_\theta^2\mathbf{I}).$$

where μ_θ and $\sigma_\theta^2\mathbf{I}$ are the part of the decoder parameters $\boldsymbol{l}$ associated to the data-based representation. In particular if we partition $\boldsymbol{l} = [\boldsymbol{l}_d, \boldsymbol{l}_g]$ we have that $\boldsymbol{l}_d = \{\mu_\theta, \sigma_\theta^2\}$. The part of the likelihood that corresponds to the grammar-based representation factorizes across all the kernels and operations, which are modelled by Multinomial distributions $Multi(n, \boldsymbol{\pi})$. In particular, denote by $\mathbf{x}_{k_j}$ and $\mathbf{x}_{o_j}$ the one-hot representation of the j -th added kernel and operation. We write the grammar-based likelihood of the decoder as

$$p_\theta(\mathbf{x}_g|\mathbf{z}) = p_\theta(\mathbf{x}_{k_1}|\mathbf{z}) \prod_{j=1}^{N_{max}-1} p_\theta(\mathbf{x}_{o_j}|\bar{\mathbf{x}}_{o_j}, \mathbf{z})p_\theta(\mathbf{x}_{k_{j+1}}|\bar{\mathbf{x}}_{o_{j+1}}, \mathbf{z})$$

where $\bar{\mathbf{x}}_{o_j} = \{\mathbf{x}_{o_1}, ..., \mathbf{x}_{o_{j-1}}\}$,

$$\mathbf{x}_{o_j}|\bar{\mathbf{x}}_{o_j}, \mathbf{z} \sim \begin{cases} Multi(1, \boldsymbol{\pi}_{o_j}) & \text{if none in } \bar{\mathbf{x}}_{o_j} \text{ is Stop} \\ 1 & \text{otherwise,} \end{cases}$$

for $j = 1, \ldots, N_{max}$ and

$$\mathbf{x}_{k_j}|\bar{\mathbf{x}}_{o_j}, \mathbf{z} \sim \begin{cases} Multi(1, \boldsymbol{\pi}_{k_j}) & \text{if none in } \bar{\mathbf{x}}_{o_j} \text{ is Stop} \\ 1 & \text{otherwise,} \end{cases}$$

for $j = 2, \ldots, N_{max}$. The next kernel and operation depend on the previous operation only, because whenever the previous operation is Stop, we ignore the kernels and operations after that. The vector parameters $\boldsymbol{\pi}_{o_j}$ and $\boldsymbol{\pi}_{k_j}$ of the multinomial are computed as the normalized versions of the corresponding blocks of $\boldsymbol{l}_g$. Note that once the *Stop* operation has been observed, the remaining $p_\theta(\mathbf{x}_{k_j}|\mathbf{z})$ and $p_\theta(\mathbf{x}_{o_j}|\mathbf{z})$ are set to one. In this way, there exists a conditional dependency of each kernel/operation on the previous factor, which refers to the appearance of *Stop* operation.

Interestingly, the SG-VAE always learns vectors that directly map to feasible kernel combinations produced by the grammar (see Figure 3.2). Note also that this model is applicable in any scenario where a context-free grammar is available. The data-based representation can also be easily dropped if not needed in other domains.

3.3.4 SG-VAE Based Search

Denote by $\mathcal{M}_{\mathbf{x}}$ the model configuration associated to the representation $\mathbf{x}$. For simplicity in the notation, in this section we will drop the dependence of the model on the parameters θ. We denote by $\tilde{\mathbf{x}}|\mathbf{z}^\star$ the mode of the decoder p_θ at $\mathbf{z}^\star$. Also, let us denote by $\mathbb{Z}$ the latent space learned by the pre-trained SG-VAE. We reformulate Problem (3.1) as finding

$$\mathbf{z}^{opt} = \arg\max_{\mathbf{z}^\star \in \mathbb{Z}} \mathcal{C}(\mathcal{M}_{\tilde{\mathbf{x}}|\mathbf{z}^\star}). \tag{3.2}$$

The hypothesis is that points $\mathbf{z}^\star$ in the latent space, and models represented by the corresponding modes $\tilde{\mathbf{x}}|\mathbf{z}^\star$, are well mapped to each other, as it is shown in Figure 3.2 in Section 3.3.3. Therefore it is possible to search for the best model in $\mathbb{Z}$ rather than in the original space.

To find $\mathbf{z}^{opt}$ we use Bayesian optimization [Shahriari et al., 2016]. We select a series of locations $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{iter}}$ such that the maximum of $\mathcal{C}$ is evaluated as quickly as possible. Following the standard practice in BO, we use a Gaussian process $p(f) = \mathcal{GP}(\mu; k)$ with mean function μ and positive-definite kernel k to model the underlying objective function, now defined between $\mathbb{Z}$ and the domain of the model selection criterion. In standard cases, the inputs of the GP are also the inputs of the objective function ($\mathcal{C}(\mathcal{M})$ in AS). In SVO, the inputs of the GP are the latent representations $\mathbf{z}$, which are different from the inputs of the objective function. This gives rise to a problem: a model configuration $\mathbf{x}$ may correspond to multiple points $\mathbf{z}$ in latent space. This uncertainty can be captured by the posterior distribution $p(\mathbf{z}|\mathbf{x})$, which is intractable in SG-VAE. Instead, we estimate a mean-field variational approximation $q_\gamma(\mathbf{z}^*)$ of the posterior detailed in Section 3.3.4.1.

3.3.4.1 Variational Posterior for Test Points

To quantify the uncertainty of the latent variable $\mathbf{z}^\star$ of test points, we propose a separate mean-field variational inference, where the posterior distribution is parameterized as $q_\gamma(\mathbf{z}^\star) = \mathcal{N}(\mathbf{z}^\star : \mu_\gamma, \sigma_\gamma^2\mathbf{I})$. This estimation of $q_\gamma(\mathbf{z}^\star)$ is done by minimizing the Kullback-Leibler divergence $\mathrm{KL}(q_\gamma(\mathbf{z}^\star)\|p(\mathbf{z}|\mathbf{x}))$ where $p(\mathbf{z}|\mathbf{x})$ denotes the true posterior of $\mathbf{z}$. We assume the $q_\gamma(\mathbf{z}^\star)$ is a Gaussian distribution with variational mean parameter μ_γ and variational variance parameter σ_γ^2. There are mainly two reasons why this is necessary to quantify the uncertainty. First, we have fixed σ_{en}^2 to be constant because it makes the training easier. Therefore, it is kept constant across the latent space and one would expect a different level of uncertainty for different test points. Second, if a

test point $\mathbf{x}^\star$ is dissimilar to any of the training points, there is no guarantee of good predictions by the encoder.

To factor in this uncertainty in the inference of GP, we use Gaussian process with uncertain inputs [Damianou et al., 2016], in which we fit a Gaussian process between inputs following the distribution $q_\gamma(\mathbf{z}^*) \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma^2 \mathbf{I})$. The idea of GP with uncertain inputs is that if the inputs z to the GP are random Gaussian variables: $p(z) = \mathcal{N}(\mu, \Sigma)$, then the prediction at a new input z^* can be found using a simple Monte-Carlo approach:

$$p(f(z^*)|\mu, \Sigma) = \int p(f(z^*)|z^*)p(z^*)dz^* \approx \frac{1}{N}\sum_{n=1}^{N} p(f(z^*)|z_n^*), \tag{3.3}$$

where z_n^* are samples from $p(z^*)$.

To sum up, for any new latent z^*, the procedure works as follows:

- Find x^* greedily, by sequentially choosing the kernels and operations with highest logits.

- Evaluate objective function $f(x^*)$.

- Propagate uncertainties into the search using a separate mean-field variational inference with $q_\gamma(\mathbf{z}^\star)$ to learn its mean and variant parameters μ_γ and σ_γ^2.

- BO is performed on tuples of $(\mu_\gamma, \sigma_\gamma^2, f(x^*))$, where σ_γ^2 is used to provide uncertain inputs to the Gaussian process.

The posterior of the GP is used to create an acquisition function that is used to select the next points to evaluate. In this work we used the Expected Improvement [Mockus, 1977] (EI) but other acquisition functions are also possible. Note that, as we use mean and variance from the GP with uncertain inputs, the distribution $q_\gamma(\mathbf{z}^\star)$ is automatically pushed into the acquisition. The next evaluation is placed at the global maximum of the EI function [Shahriari et al., 2016]. See Algorithm 2 for a full description of the algorithm that we call *Structured Variationally Auto-encoded Optimization* (SVO).

3.4 Experiments

We show three experiments. The first one explains the behaviour of the method in a time series. The goal is to interpret how new kernels are selected as soon as we provide more data to the system so that more structure in the kernel can be inferred. The second experiment compares SVO with some baselines. The third one formulates natural scene understanding as a search problem in a structured space and apply SVO to infer the content in a natural scene. Inspired by [Wu et al., 2017], we use "Minecraft" as a natural scene generation engine and show SVO successfully produced good interpretation of images with a few attempts.

3.4.1 The Airline Dataset

We apply the SVO algorithm to fit the Airline passenger data [Box and Jenkins, 1990], a time series that consists of 144 monthly totals of airline passengers from January 1949 to December 1960. We consider 4 base kernels: squared exponential (SE), linear (LIN), periodic (PER) and rational quadratic (RQ) and we optimize the mean squared error (MSE) of the prediction. We use uniform prior probabilities in both the grammar operations and the base kernels. We allow a maximum of 5 operations in the grammar generating process. We train a SG-VAE with 2 hidden layers with 400 hidden units on $20,000$ simulated kernel combinations (from which 4697 where unique) and 5 dimensions in the latent space. We apply Algorithm 2 using the first 10%, 37% and 63% of the data. The rest is used for testing.

In Figure 3.3 we show the fit of the GP models with the combinations proposed by SVO. Interestingly, with 10% of the data the seasonal structure is captured. The solution found is $SE \times PER + RQ + PER$. With 37% of the data the solution found is $PER \times PER + PER + RQ + LIN$, which is capturing the linear trend. With 63% of the data the best solution found is $PER \times SE \times RQ \times LIN + SE$. In this case, the method is capturing not only the linear trend but also the interactions with the seasonal components. SVO is able to extract coherent structure in the data.

3.4.2 Comparison Across Different Optimization approaches

We do optimal model selection with GPs in three different datasets: the Airline dataset described in the previous section and the Barley and Concrete datasetsd[2]. The concrete dataset has dimension 8, while the rest are unidimensional. The experimental

[2]All datasets are available at the UCI repository: https://archive.ics.uci.edu/ml/index.php except the Barley, which can be found at https://datamarket.com

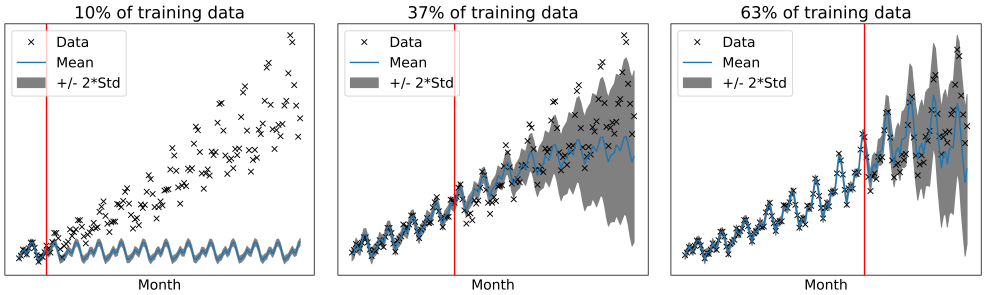


Figure 3.3: Results obtained by the SVO method in the Airline dataset using four base kernels. We search for the optimal combination with different sample sizes. The vertical red lines of the plots represent the separation between training data (on the left of the line) and test data (on the right).

set up is the same as the one we used in section 3.4.1. We compare SVO with different configurations. We learn the SG-VAE using both the grammar data and the distance representations described in Section 3.3.2. To generate the distances, the hyper parameters of the base kernels are obtained after maximizing the marginal log-likelihood of the models. The parameters of the kernel combinations are sampled from hyper-priors that were specified as in [Malkomes et al., 2016]. In the search, we include the case in which the uncertainty of the latent space of the SG-VAE is used using Section 3.3.4.1 and the case in which no uncertainty is propagated (just $\mathbf{z}^\star$ is used in the search). We compare different versions of the SVO (with uncertainty and without uncertainty) with (i) *Compositional Kernel Search* (CKS) [Duvenaud et al., 2013], (ii) a random search carried out in the original kernel configurations space, (iii) a random search carried out in the latent space of the SG-VAE learned in each scenario and (iv) standard BO. The vanilla BO is performed on 12 categorical variables: one of them represents the number of operations of the combination which takes values in $\{1, 2, 3, 4, 5, 6\}$; 6 of them are categorical variables with 4 categories representing the kernels and 5 of them are categorical variables with 2 categories representing the operations (addition or multiplication). A detailed description of each of the baselines can be found in the appendix.

In all cases we run the optimization for 70 iterations. In Figure 3.4 we show the best current MSE found in each iteration, which has been averaged over 5 runs for different initial points.

The CKS needs to compose complex kernels from single ones, which makes its convergence slow (it always starts with one kernel and sequentially composes more complex ones). That is the reason why it gives worse performance than vanilla BO and random search on simpler 1D examples. In the other methods, this is not needed

as complex combinations can be selected from the beginning of the optimization. This is why the SVO convergence curves are flatter, and SVO also converges faster and to a better optimum compared to CKS in the three datasets. Propagating the uncertainty of the SG-VAE has a positive effect on the search in the Barley dataset.

3.4.3 Natural Scene Understanding

Understanding natural scene is a challenging task because of the almost infinite number of possible combinations of objects and large variation of their appearance on images. On the other hand, we have sophisticated graphics rendering engines. We use computer graphics engine to improve scene understanding by formulating natural scene understanding as a search problem, i.e., given a natural image, whether we can find a configuration of a graphics rendering engine to reproduce the query.

Following the experiment setting of [Wu et al., 2017], we take "Minecraft" as the rendering engine and use XML as the description of configuration. We consider 12 Minecraft objects such as pig, cow, sheep and chicken, etc., and consider two attributes of each object, i.e., the location of an object on the image parametrized by the distance to the camera and the 1D angle from right to left. Both attributes are discretized into five different categories. The task is to search for the XML configuration of which the generated image matches the target image as close as possible. For simplicity, we use the $L2$ norm to measure the dissimilarity between a generated image and the target image, but more sophisticated dissimilarity measure can be used here.

The target images are shown in Figure 3.5a, 3.5b, 3.5c. We apply SVO to this problem by defining a structured generation process like the one in Section 3.3.3. We consider two types of operations: adding a new object and stop. When adding a new object, we choose its object type and the values of two attributes. With this generation process, we generated 20,000 configurations for training the SG-VAE. We use the grammar-based representation for simplicity. Then, we apply the search algorithm described in Section 3.3.4 to look for the best configuration. After 100 iterations, the image generated by the best configuration is shown in Figure 3.5d, 3.5e, 3.5f. The generated images closely match with the target ones with small disparity. Note that our approach is significantly different from [Wu et al., 2017]. Wu et al. [2017] trained neural networks to directly predict a configuration given a target image. This requires generation of tens of thousands of images from the rendering engine for training the neural networks. The number of required training images can easily go a few magnitudes higher if more objects or attributes are considered. SVO is a meta-approach for this problem, where we *do not* learn a model to directly analyse

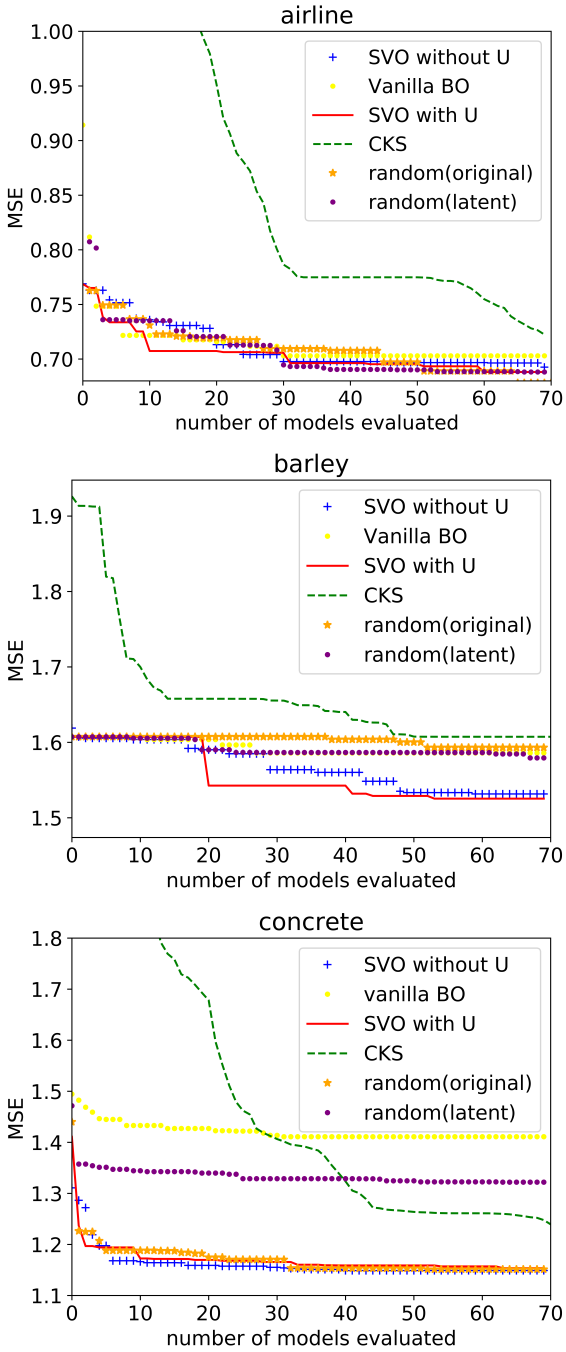


Figure 3.4: Comparison of the SVO algorithm and the Compositional Kernel Search (CKS). SVO is able to optimize the model selection criterion faster in all datasets. Interestingly, SVO does not need to start the search on the base kernels which allows it to consider more complex combinations faster that CKS.

images. Instead, we build a SG-VAE for configurations and search in the latent space of configurations for a matching explanation of the target image. Our approach only requires generating a few images to guide the search.

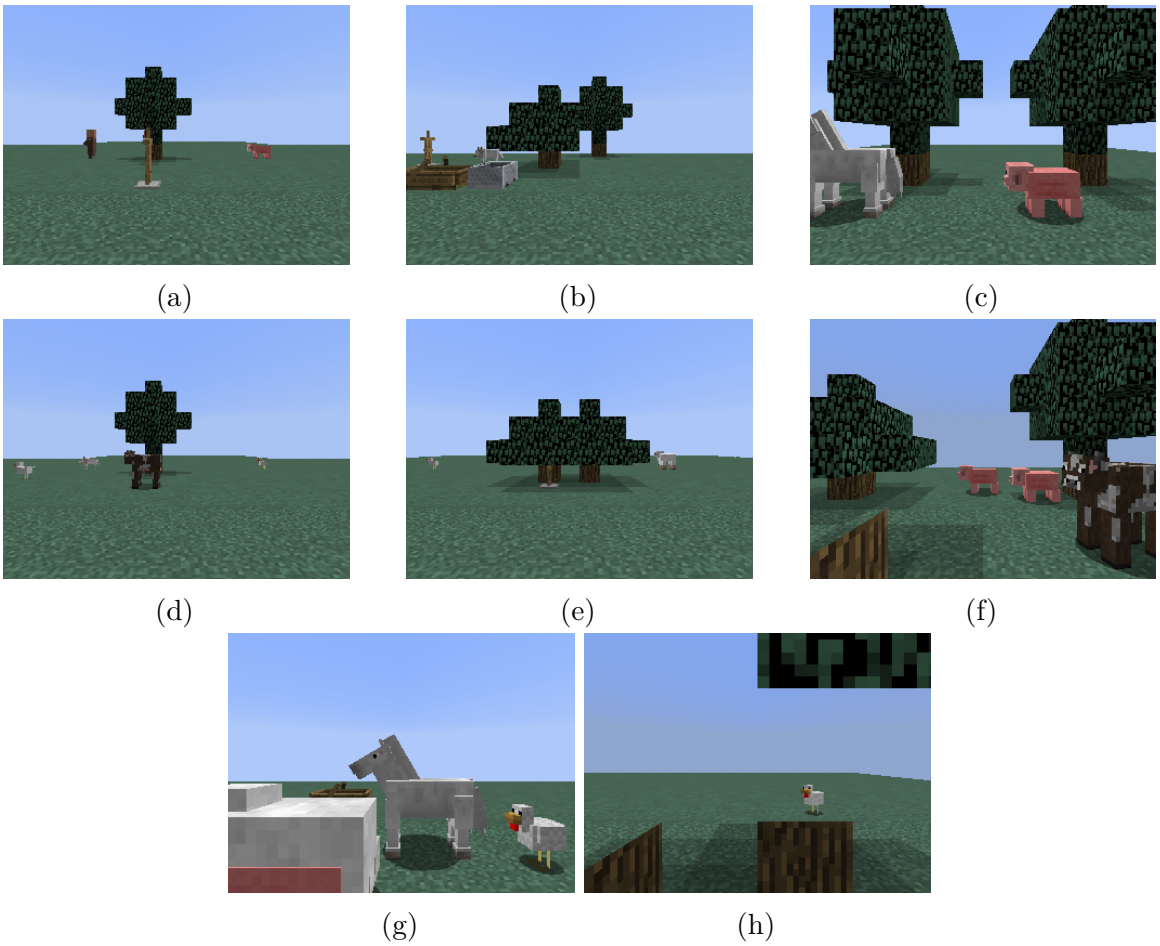


Figure 3.5: Use SVO to search for a XML configuration of the "Minecraft" engine to reproduce three target images (a), (b) and (c). The best configuration found by SVO are (d), (e) and (f) respectively. Images (g) and (h) were selected randomly to illustrate the complexity of the problem.

3.5 Conclusions and Future Work

We have presented a new algorithm, SVO, for performing optimization in highly structured spaces. The idea of SVO is to learn a probabilistic low-dimensional latent space in which the solution of some high dimensional optimization problem can be easily found and mapped back to the original problem. This process is not possible without extra assumptions, which in our case are imposed by means of a context-free grammar. The SG-VAE, a new Variational Auto-encoder, explicitly uses the structure of the problem to generate feasible solutions. Bayesian optimization ideas are used in the latent space learned by the SG-VAE. The uncertainty in the latent space is propagated into the optimization using variational inference and a Gaussian process with uncertain inputs. The results obtained in the experimental section are competitive. However, we believe that there is still a large room for improvement in this research direction. For example, we see a limitation in the use of VAE as it is not well suited to learn the dimension of the latent space from the data. A GP-LVM could be a possible alternative. Other limitations of the SG-VAE are related to smoothness and structure of the problem. Even if the dimensionality of the original space is large the method is expected to work if it is highly structured. However, in low dimensional spaces with not mush structure the SG-VAE is not expected to be beneficial (information cannot be compressed). It is also interesting to investigate how previous knowledge can be embedded in the search. In this work we used a context-free grammar to favour certain solutions of the input space, similarly to the idea of using transfer learning when similar problems have been solved before. Further combinations of this ideas will be explored in the future.

3.6 Appendix

3.6.1 Visualization and Interpretation of the Latent Space of the SG-VAE in the Airline Dataset

We display the learned latent space of the SG-VAE for the airline dataset in Figure 3.6. Each dot represents a kernel combination learned by the VAE in the projected 2-dimensional latent space. We can observe the effect of the data-based representation to the latent space that is able to distinguish between complicated kernel combinations.

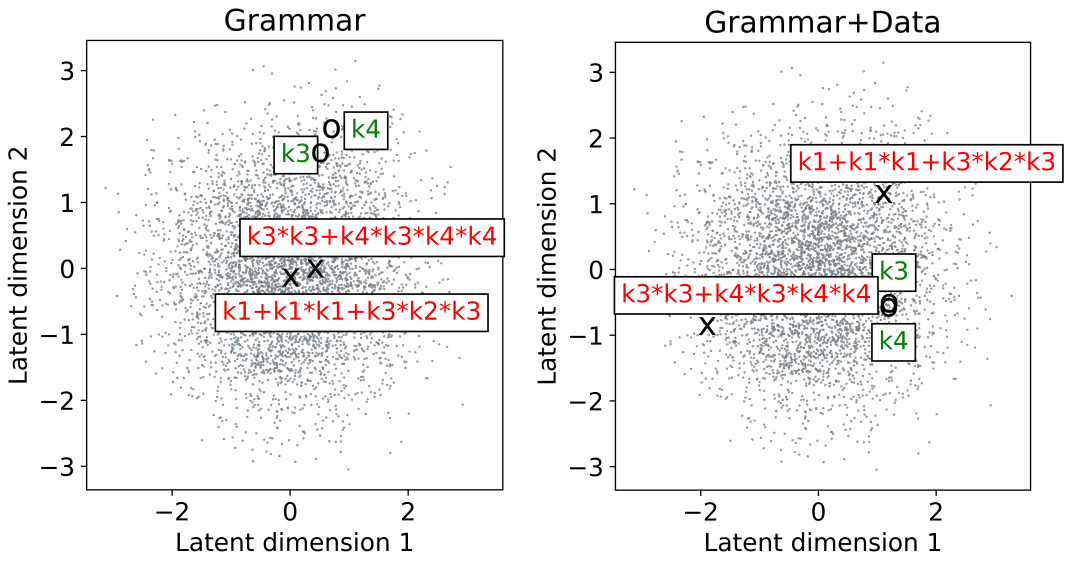


Figure 3.6: Visualization of the latent space for the airline dataset. This is a 2D projection of the 5D latent space (a): latent space with the grammar kernel representation, (b): latent space with the grammar + data-based kernel representation.

3.6.2 Experimental Set Up Details

For each dataset in the automatic statistician experiments in Section 3, we evenly split the dataset into a training and a test set and use the test Mean Square Error (MSE) as the model selection criterion.

We used the same configuration of Structure Generating Variational Auto-encoder (SG-VAE) for the automatic statistician experiments. The configuration is as follows:

- Use two hidden layers with 400 hidden units in both the encoder and the decoder
- Use a five dimensional latent space.
- Use Adam as the optimizer with learning rate equals to 0.005.

The details of search in latent space is as follows:

- Use squared exponential kernel in the Gaussian Process to model the relationship between the model criterion and the latent space, with the acquisition function being the expected improvement.
- Use GPyOpt `http://github.com/SheffieldML/GPyOpt` as the tool to do the optimization.

3.6.3 Description of the Baselines in the Comparison

We have compared SVO with others baselines in the experiment section, namely the Vanilla BO, random search in the original space and random search in the latent space. We hereby give the details of those methods.

Although vanilla BO is not well suited for the problems in our work since the input space in our experiments is categorical (types of kernels and operations) and hierarchical (the number of operations in the combination affects the length of the combination), which is one of the main motivations of our paper, we have included it as a baseline comparison. There are 12 variables, with one of them being the variable representing the number of operations of the combination which takes values in $\{1, 2, 3, 4, 5, 6\}$, 6 of them being categorical variables with 4 categories representing the kernels , and 5 of them being categorical variables with 2 categories representing the operations (addition or multiplication). The hyper-priors for sampling the parameters of the kernel combinations and the setup of the BO search is the same as SVO.

We also compared our method with random search, both in the original and the latent space. For random (original), we randomly select a kernel combination from the generated grammar representation; for random (latent), we randomly sample from the latent space and use the decoder of the learned VAE to decode into a feasible kernel combination.

Chapter 4

Relativistic Hamiltonian Monte Carlo

This chapter is based on the paper

- **Xiaoyu Lu**, Valerio Perrone, Leonard Hasenclever, Sebastian J Vollmer and Yee Whye Teh.
  Relativistic Monte Carlo
  https://arxiv.org/abs/1609.04388.
  International Conference on Artificial Intelligence and Statistics, 2017.

The contributions I have made to this paper includes:

- The derivation and implementation of the RSGD algorithm.
- The derivation and implementation of the RSGNHT algorithm.
- Experiments on the Banana, logistic regression, MNIST examples.

4.1 Abstract

Hamiltonian Monte Carlo (HMC) is a popular Markov chain Monte Carlo (MCMC) algorithm that generates proposals for a Metropolis-Hastings algorithm by simulating the dynamics of a Hamiltonian system. However, HMC is sensitive to large time discretizations and performs poorly if there is a mismatch between the spatial geometry of the target distribution and the scales of the momentum distribution. In particular the mass matrix of HMC is hard to tune well. In order to alleviate these problems we propose relativistic Hamiltonian Monte Carlo, a version of HMC based on relativistic dynamics that introduce a maximum velocity on particles. We also derive

stochastic gradient versions of the algorithm and show that the resulting algorithms bear interesting relationships to gradient clipping, RMSprop, Adagrad and Adam, popular optimisation methods in deep learning. Based on this, we develop relativistic stochastic gradient descent by taking the zero-temperature limit of relativistic stochastic gradient Hamiltonian Monte Carlo. In experiments we show that the relativistic algorithms perform better than classical Newtonian variants and Adam.

4.2 Introduction

In Bayesian statistics, Markov chain Monte Carlo (MCMC) [Metropolis et al., 1953] techniques have been popular tools to simulate from the posterior distributions. However, traditional MCMC methods suffer from slow mixing and high correlation especially in high-dimensional models, due to the random walk proposals. An alternative method, namely Hamiltonian Monte Carlo (HMC) [Duane et al., 1987, Neal, 2010, Carpenter, 2015, Hoffman and Gelman, 2014], is based on the Hamiltonian dynamics of molecules. The Hamiltonian consists of a potential energy function which is usually represented as the target posterior distribution, and a kinetic energy of the molecules. The molecules follow Newton's laws of motion to explore the energy landscape, thus are able to explore distant regions and reduce the random walk effect. Another important advantage of Hamiltonian dynamics is that it preserves volume, and therefore avoids the problem of computing the possibly intractable determinant of the Jacobian matrix when calculating the acceptance probability of Metropolis updates.

On the other hand, the gain in efficiency of exploration often comes with a cost in computation time for large datasets, where the computational burden lies primarily on the evaluation of the gradients of the log density. In order to cope with scalability, various stochastic gradient variants based on diffusive dynamics [Welling and Teh, 2011a, Ma et al., 2015, Ding et al., 2014, Chen et al., 2014a] have been developed, where instead of using the full dataset, only a subset of dataset (called minibatches) is used to calculate the gradients.

In this work, we consider replacing the Newton's laws of motion of particles with Einstein's relativity of photons, so that the speed of the particles are upper bounded by the speed of light. We will show experimentally that the new algorithms are more robust than the classical Newtonian versions and allow for larger time discretisations of the dynamical systems. We will also show the connection of the resulting algorithms with various optimisation algorithms including RMSprop, Adagrad and Adam[Kingma

and Ba, 2014a] which are popular in deep learning. We then develop the stochastic variants of the algorithm, including the introduction of the Nosé-Hoover thermostats. Finally, by taking the limit as the temperature goes to zero in the *stochastic gradient relativistic Hamiltonian Monte Carlo (SGHMC)* algorithm, we propose a new stochastic gradient descent algorithm called the *Relativistic Stochastic Gradient Descent (RSGD)* that outperforms the state-of-the-art Adam on neural network examples.

4.3 Background

Suppose we are interested in sampling from the posterior probability density $p(\theta|\mathbf{X}) \propto p(\mathbf{X}|\theta)p(\theta)$, where $\theta \in \mathbf{R}^n$ denotes the parameter and $\mathbf{X}$ denotes the data. We can write the density in its canonical form as $p(\theta|\mathbf{X}) \propto \exp(-U(\theta))$, where $U(\theta) = -\log p(\mathbf{X}|\theta) - \log p(\theta)$ is called the potential energy. The classical HMC algorithm, which we refer to as *Newtonian Hamiltonian Monte Carlo (NHMC)* augments the space of parameters with auxiliary variables p, and we regard θ and p as the position and momenta of particles in a system. The joint density is then $p(\theta, p) \propto \exp(-H(\theta, p))$, where $H(\theta, p) = U(\theta) + K(p)$ is called the *Hamiltonian*, U is the potential energy, and $K(p) = \frac{1}{2m}p^T p$ is the kinetic energy with mass m. The Hamilton's equations are written as

$$\dot{\theta} = \frac{\partial H}{\partial p}, \qquad \dot{p} = -\frac{\partial H}{\partial \theta} \tag{4.1}$$

Plugging H into 4.1 gives

$$\dot{\theta} = \nabla K(p), \qquad \dot{p} = -\nabla U(\theta) =: f(\theta) \tag{4.2}$$

where $f(\theta) := -\nabla U(\theta)$ is defined as the force on the system.

In order to implement the algorithm on a computer, we have to approximate the Hamilton's equation by discretizing time using small increments ϵ. Here we adopt the *leapfrog method* [Calvo, 1994], which is a modification of Euler's method that preserves volume exactly and gives better results. The algorithm starts with a half step update for the momentum variables, then a full step update for the position variables with the new values of the momentum variables, and finally another half step update for the momentum variables, using the updated values of the position variables. The algorithm is summarized below:

$$p_{t+\frac{\epsilon}{2}} = p_t - \frac{\epsilon}{2}\nabla U(\theta_t) \tag{4.3}$$

$$\theta_{t+\epsilon} = \theta_t + \frac{\epsilon}{m}p_{t+\frac{\epsilon}{2}} \tag{4.4}$$

$$p_{t+\epsilon} = p_{t+\frac{\epsilon}{2}} - \frac{\epsilon}{2}\nabla U(\theta_{t+\epsilon}) \tag{4.5}$$

Therefore given current states (θ, p), one proposes new states (θ^*, p^*) by simulating the Hamiltonian dynamics for L leapfrog steps, then a Metropolis Hasting step is performed to correct for the discretization error, with acceptance probability given by

$$\min[1, \exp(-H(\theta^*, p^*) + H(\theta, p))]$$

It can be seen that the velocity of the particles reads $\frac{\epsilon}{m}p_{t+\frac{\epsilon}{2}}$, so if the mass m is too small, the particles may travel too fast and miss the mode of the posterior density, therefore a small ϵ is needed to compensate for the discretization error, which increases the computation cost. On the other hand, too small values of ϵ results in slow mixing of the chain. This motivated our research to have more control over the velocity of the particles and develop a more robust algorithm. We propose the Relativistic Monte Carlo methods with Einstein's relativistic kinetic energy such that the speed of the particles is limited by the speed of light.

4.4 Relativistic Hamiltonian Monte Carlo

We replace the Newtonian kinetic energy K with Einstein's special relativity[Einstein, 1905] in the Hamiltonian:

$$H(\theta, p) = U(\theta) + K(p) \tag{4.6}$$

$$K(p) = mc^2\left(\frac{p^\top p}{m^2c^2} + 1\right)^{\frac{1}{2}}, \tag{4.7}$$

where the target density is $p(\theta) \propto \exp(-U(\theta))$, and $\theta, p \in \mathbb{R}^d$ represent the position and momenta of the particles respectively as before. m is the "rest mass " and c is the "speed of light". Note the marginal distribution of the momenta is $p(p) \propto \exp(-K(p))$ which is a multivariate generalisation of the hyperbolic distribution as opposed to the Gaussian distribution in the Newtonian case.

Applying Hamilton's equation (4.1) gives

$$\dot{\theta} = \frac{\partial H}{\partial p} = M^{-1}(p)p, \qquad \text{with} \quad M(p) = m\left(\frac{p^\top p}{m^2c^2} + 1\right)^{\frac{1}{2}}$$
$$\dot{p} = -\frac{\partial H}{\partial \theta} = -\nabla U(\theta). \tag{4.8}$$

We can interpret the scalar $M(p)$ as the relativistic mass and note the speed is now

$$\|M^{-1}(p)p\| = \frac{c\|p\|}{\sqrt{p^T p + mc^2}} \leq c,$$

which is limited by the speed of light, and the smaller the rest mass m, the bigger the "cruising " speed, and as the m goes to zero, the particles travel at the speed of light acting like a photon. This gives a good intuition of the tuning on the hyperparameters based on the target density, where we limit the speed of the parameters by c and use m to control the "cruising " speed. More details and illustration of hyperparameter tuning can be found in the experimental Section 4.7.

Note that all the coordinates of the momenta have the same relativistic mass $M(p)$, but parameters in different dimensions might want to behave differently, some coordinates might desire to move faster while others prefer to slow down. Setting $M(p)$ to be the same might lead to slow mixing in some dimensions or instability caused by too large velocity in other dimensions. To be able to control the dimensions separately, we could separate the kinetic energy as a sum of kinetic energies in each coordinate, with possibly different mass m_j and speed of light c_j:

$$K(p) = \sum_{j=1}^{d} m_j c_j^2 \left(\frac{p_j^2}{m_j^2 c_j^2} + 1\right)^{\frac{1}{2}}, \tag{4.9}$$

where j indexes the coordinate of the d-dimensional system. This leads to the same Hamiltonian dynamics (4.8), but with all variables interpreted as vectors, and all arithmetic operations interpreted as element-wise operations. For simplicity, in the following we will describe only the non-separable version (4.7).

The *Relativistic Hamiltonian Monte Carlo (RHMC)* algorithm can be summarized as follows: In each iteration, suppose the current state is (θ_0, p_0),

- Simulate proposal momenta according to $p(p) \propto \exp(-K(p))$ with $K(p)$ given in (4.7); here we used adaptive rejection sampling.

- Do L leapfrog steps to obtain (θ, p) similar to (4.5): for $t = 1, ...L$

$$\begin{aligned} p_{t+1/2} &\leftarrow p_t - \tfrac{1}{2}\epsilon \nabla U(\theta_t) \\ \theta_{t+1} &\leftarrow \theta_t + \epsilon M^{-1}(p_{t+1/2}) p_{t+1/2} \\ p_{t+1} &\leftarrow p_{t+1/2} - \tfrac{1}{2}\epsilon \nabla U(\theta_{t+1}) \end{aligned} \tag{4.10}$$

- Do a Metropolis-Hastings accept/rejection step: accept the (θ, p) with probability

$$\min[1, \exp(-H(\theta_L, p_L) + H(\theta_0, p_0)]$$

Notice although there is no limit on the magnitude of the momenta p, the "cruising speed" of the parameters θ is upper bounded by the speed of light times the stepsize, which is not the case for NHMC. When the target distribution is very peaky, the updates for θ can be so big for NHMC that makes it necessary for smaller stepsize ϵ to account for discretization error, resulting in low efficiency of the sampler. However, RHMC does not exhibit this issue and is a more stable and robust algorithm. Note the relationship of RHMC with other popular optimisation algorithms in deep learning such as Adagrad, RMSprop and Adam, where the learning rates are adapted individually in each dimension, so that the updates of the parameters have similar scales across dimensions.

4.4.1 Relativistic Stochastic Gradient Markov Chain Monte Carlo

In recent years stochastic gradient MCMC (SGMCMC) algorithms have been very well explored as methods to scale up Bayesian learning by using mini-batches of data [Welling and Teh, 2011a, Chen et al., 2014a, Ding et al., 2014, Ma et al., 2015, Shang et al., 2015]. In this section we develop relativistic variants of SGHMC [Chen et al., 2014a] and SGNHT [Ding et al., 2014, Shang et al., 2015]. These algorithms include momenta, which serve as reservoirs of previous gradient computations, thus can integrate and smooth out gradient signals from previous mini-batches of data. As noted earlier, because the momentum can be large, particularly as the stochastic gradients can have large variance, the resulting updates to θ can be overly large, and small values of the stepsize are required for stability, leading potentially to slower convergence. This motivates our development of relativistic variants.

We make use of the framework of [Ma et al., 2015] for deriving SGMCMC algorithms. Let z be a collection of variables with target distribution $f(z) \propto e^{-H(z)}$. Consider a SDE in the form

$$dz = -[D(z) + Q(z)]\nabla H(z)dt + \Gamma(z)dt + \sqrt{2D(z)}dW \quad \Gamma_i(z) = \sum_{j=1}^{d} \frac{\partial[D_{ij}(z)+Q_{ij}(z)]}{\partial z_j}, \tag{4.11}$$

where $D(z)$ is a symmetric positive-definite diffusion matrix, $Q(z)$ is a skew-symmetric matrix which describes energy-conserving dynamics, $\Gamma(z)$ is a correction factor, and W is the d-dimensional Weiner process (Brownian motion). [Ma et al., 2015] showed that under mild conditions, the SDE converges to the desired stationary distribution $f(z)$. Hence, in the following we simply have to choose the appropriate z, D and Q. Once the correction factor Γ is computed, the SDE is discretised, and a stochastic estimate $\nabla\tilde{U}(z)$ for $\nabla U(z)$ is substituted, we obtain a correct relativistic SGMCMC algorithm. The stochastic gradient has asymptotically negligible variance compared to the noise injected by W.

4.4.2 Relativistic Stochastic Gradient Hamiltonian Monte Carlo

Suppose our noisy gradient estimate $\nabla\tilde{U}(\theta)$ of $\nabla U(\theta)$ is based on a minibatch of data and, using the central limit theorem, we assume that $\nabla\tilde{U}(\theta) \approx \nabla U(\theta) + \mathcal{N}(0, B(\theta))$. We take $z = (\theta, p)$, the relativistic Hamiltonian $H(z)$(4.7), a fixed symmetric diffusion matrix D and

$$D(z) = \begin{bmatrix} 0 & 0 \\ 0 & D \end{bmatrix}, Q(z) = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix}, \text{and thus } \Gamma(z) = \mathbf{0}. \tag{4.12}$$

The resulting relativistic SGHMC dynamics becomes,

$$\begin{pmatrix} d\theta \\ dp \end{pmatrix} = \begin{pmatrix} M^{-1}(p)p \\ -\nabla U(\theta) - DM^{-1}(p)p \end{pmatrix} dt + \begin{pmatrix} 0 \\ \sqrt{2D} \end{pmatrix} dW_t$$

Using a simple Euler discretisation, the SGHMC algorithm becomes,

$$\begin{aligned} p_{t+1} &\leftarrow p_t - \epsilon_t \nabla\tilde{U}(\theta_t) - \epsilon_t DM^{-1}(p_t)p_t + \mathcal{N}(0, \epsilon_t(2D - \epsilon_t\hat{B}_t)) \\ \theta_{t+1} &\leftarrow \theta_t + \epsilon_t M^{-1}(p_{t+1})p_{t+1}, \end{aligned} \tag{4.13}$$

where $\hat{B}$ is an estimate of the noise coming from the stochastic gradient $B(\theta)$. The term $DM^{-1}(p)p$ can be interpreted as friction, which prevents energy from building up and corrects for the noise coming from the stochastic gradient.

It is useful to compare RSGHMC with preconditioned SGLD [Li et al., 2015, Teh et al., 2015] which attempts to adapt the SGLD algorithm to the geometry of the space, using adaptations similar to RMSProp, Adagrad or Adam. The relevant term above is the update $M^{-1}(p_{t+1})p_{t+1}$ to θ_{t+1}:

$$M^{-1}(p_{t+1})p_{t+1} = \frac{p_{t+1}}{\sqrt{\frac{p_{t+1}^\top p_{t+1}}{c^2} + m^2}}. \tag{4.14}$$

Note the surprising similarity to RMSProp, Adagrad and Adam, with the main difference being that the relativistic mass adaptation uses the current momentum instead of being separately estimated using the square of the gradient. This has the advantage that the relativistic SGHMC enforces a maximum speed. In contrast, preconditioned SGLD scales up small gradients while increasing the injected noise as well, which can be significantly larger than the gradient updates, making the algorithm unstable.

4.5 Relativistic Stochastic Gradient Descent (with Momentum)

Motivated by the relationship to RMSprop, Adagrad and Adam, we develop a relativistic stochastic gradient descent (RSGD) algorithm with momentum, by taking the zero-temperature limit of the RSGHMC dynamics. From thermodynamics, the canonical (Gibbs Boltzmann) density is proportional to $e^{-\beta U(z)}$ where $\beta = 1/k_B T$, k_B denotes the Boltzmann constant and T represents the temperature. Previously, we have been using $\beta = 1$ which corresponds to the posterior distribution. For an arbitrary value of β,

$$\begin{pmatrix} d\theta \\ dp \end{pmatrix} = \begin{pmatrix} \beta M^{-1}(p)p \\ \beta\left(-\nabla U(\theta) - DM^{-1}(p)p\right) \end{pmatrix} dt + \begin{pmatrix} 0 \\ \sqrt{2D} \end{pmatrix} dW. \tag{4.15}$$

By taking $\beta \to \infty$, the target distribution becomes more peaked around the MAP estimator. Simulated annealing [Geman and Hwang, 1986, Andrieu and Doucet, 2000], which increases $\beta \to \infty$ over time, forces the sampler to converge to a MAP estimator. Instead, we can derive RSGD by rescaling time as well, $\hat{\theta}(t) = \theta(\beta t)$, $\hat{p}(t) = p(\beta t)$,

$$\begin{pmatrix} d\hat{\theta} \\ d\hat{p} \end{pmatrix} = \begin{pmatrix} M^{-1}(\hat{p})\hat{p} \\ -\nabla U(\hat{\theta}) - DM^{-1}(\hat{p})\hat{p} \end{pmatrix} dt + \begin{pmatrix} 0 \\ \sqrt{\frac{2D}{\beta}} \end{pmatrix} dW \tag{4.16}$$

and letting $\beta \to \infty$, obtaining the following ODE:

$$\begin{pmatrix} d\theta \\ dp \end{pmatrix} = \begin{pmatrix} M^{-1}(p)p \\ -\nabla U(\theta) - DM^{-1}(p)p \end{pmatrix} dt. \tag{4.17}$$

Discretising the above system gives RSGD. Notice that if the above system converges, i.e. $\dot{\theta} = \dot{p} = 0$, it does so at a critical point of U. Similar to other adaptation schemes, RSGD adaptively rescales the learning rates for different parameters, which enables effective learning especially in high dimensional settings. Moreover, the update in each iteration is upper bounded by the speed of light. Our algorithm differs from others through the use of a momentum, and adapting based on the momentum instead of the average of squared gradients.

4.6 A Stochastic Gradient Nosé-Hoover Thermostat for Relativistic Hamiltonian Monte Carlo

In order to remedy the problem of the unknown noise coming from the stochastic gradients, a Nosé-Hoover [Leimkuhler and Reich, 2009] *thermostat* ξ in statistical physics is introduced to stabilize momentum fluctuations. The dynamics for the Newtonian stochastic gradient Nosé-Hoover thermostat (NSGNHT)[Ding et al., 2014] can be expressed as:

$$\begin{aligned} d\theta &= pdt \\ dp &= -\nabla U(\theta)dt - \xi pidt + \sqrt{2D}\mathcal{N}(0, dt) \\ d\xi &= \frac{1}{d}\left(p^T p - d\right)dt. \end{aligned}$$

The idea is that if the mean kinetic energy is big, then ξ gets bigger to 'heat' the system, so p experiences more friction; on the other hand, if the mean kinetic energy is small, then ξ gets smaller to 'cool down' the system and p experiences less friction. Intuitively, the dynamic for ξ can be motivated by the fact that at equilibrium we have

$$\mathbb{E}\left[\frac{\partial^2 K}{\partial p_i^2}\right] = \int \frac{\partial^2 K}{\partial p_i^2} e^{-K(p)} dp = -\int \frac{\partial K}{\partial p_i}\left(-\frac{\partial K}{\partial p_i} e^{-K(p)}\right) dp = \mathbb{E}\left[\left(\frac{\partial K}{\partial p_i}\right)^2\right],$$

and hence $\mathbb{E}\left[\frac{d\xi}{dt}\right] = 0$. The additional dynamic pushes the system towards $\frac{d\xi}{dt} = 0$ suggesting that the distribution will be pushed closer to the equilibrium. This gives a recipe for a stochastic gradient Nosé-Hoover thermostat with a general kinetic energy $K(p)$. We are now in the position to derive the SDE preserving the probability density

$\propto \exp(-H)$ by adopting the framework of [Ma et al., 2015] and defining:

$$H(\theta, p, \xi) = U(\theta) + K(p) + \frac{d}{2}(\xi - D)^2 \tag{4.18}$$

$$D(\theta, p, \xi) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & D \cdot I & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.19}$$

$$Q(\theta, p, \xi) = \begin{pmatrix} 0 & -I & 0 \\ I & 0 & \nabla K(p)/d \\ 0 & -\nabla K(p)^T/d & 0 \end{pmatrix}. \tag{4.20}$$

From (4.11) it follows that $\Gamma = (0\ 0\ -\Delta K(p)/d)^T$ and the dynamics become

$$\begin{pmatrix} d\theta \\ dp \\ d\xi \end{pmatrix} = \begin{pmatrix} \nabla K(p) \\ -\nabla \tilde{U} dt - \xi \nabla K(p) \\ \frac{1}{d}\left(\|\nabla K(p)\|^2 - \Delta K(p)\right) \end{pmatrix} dt + \begin{pmatrix} 0 \\ \sqrt{2D} \\ 0 \end{pmatrix} dW_t,$$

where Δ is the Laplace operator defined as $\Delta K(p) = \sum_i \frac{\partial^2 K(p)}{\partial p_i^2}$. For the relativistic kinetic energy $K(p)$, we have $\nabla_p K(p) = M^{-1}(p)p$ with $M(p) := m\left(\frac{p^T p}{m^2 c^2} + 1\right)^{\frac{1}{2}}$ a scalar and $\Delta K(p) = tr\left(\frac{d}{dp}\left(\frac{1}{d} M^{-1}(p)p\right)\right)$. The Stochastic Gradient Nosé-Hoover thermostat for relativistic HMC follows:

$$\begin{pmatrix} d\theta \\ dp \\ d\xi \end{pmatrix} = \begin{pmatrix} M^{-1}(p)p dt \\ -\nabla \tilde{U} - \xi M^{-1}(p)p \\ \frac{p^T p}{d}\left(M^{-2}(p) + c^{-2} M^{-3}(p)\right) - M^{-1}(p) \end{pmatrix} dt + \begin{pmatrix} 0 \\ \sqrt{2D} \\ 0 \end{pmatrix} dW_t.$$

4.7 Experiments

4.7.1 Small Examples

We first explore the performances of the algorithms on toy examples including a two dimensional banana function (Banana) [Haario et al., 1999] with density $p(\mathbf{x}) \propto \exp\{-0.5(0.01x_1^2 + (x_2 + 0.1x_1^2 - 10)^2)\}$, and Gaussian mixture models (GMM1, GMM2, GMM3) obtained by combining the three following Gaussian random variables with equal mixing proportions: $\mathcal{N}(-5, 1/\sigma^2)$, $\mathcal{N}(0, \sigma^2)$, $\mathcal{N}(5, 1/\sigma^2)$, where $\sigma^2 = 1, 0.5, 0.3$. When $\sigma^2 = 1$ the three Gaussians have the same variance and lower σ^2 means larger discrepancies among their variances and thus a wider range of length scales and log density gradients. The density plots of the examples can be found in the top row of Figure 4.3.

We start with an exploration of the behaviour of RHMC as the tuning parameters m, c and ϵ are varied. Firstly we consider the effective sample sizes (ESS) of the

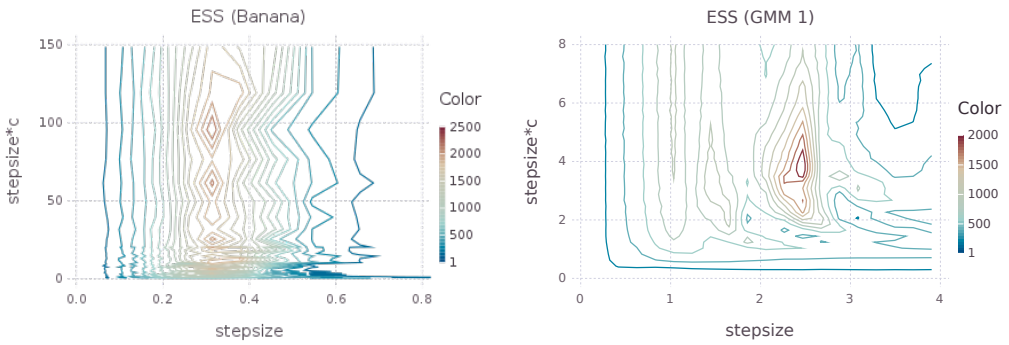


Figure 4.1: ESS contour plots of $\epsilon \times c$ versus ϵ for Banana (left) and GMM1 (right) datasets.

algorithm on the Banana and GMM1 datasets. We varied both ϵ and $\epsilon \times c$ over a grid, and computed the average ESS, over 20 chains, each of length 10^4 for Banana, and over 100 chains of length 10^5 for GMM1. The ESS contour plots can be found in Figure 4.1, which suggests that ϵc and ϵ can be independently tuned. While ϵ controls the time discretisation of the continuous-time dynamics, ϵc controls the maximum change in the parameters at each leapfrog step. Next we varied the mass parameter m for GMM1, showing plots in Figure 4.2. As expected the ESS is optimised at an intermediate value of m, and the average "cruising speed" $\bar{v}$ decreases with m. In order to understand how to tune m, on the fourth panel we overlaid two contour plots: one for ESS and the other for $\bar{v}$. We see that the cruising speed $\bar{v}$ correlates much more with the ESS than m does. This suggests that m should be tuned via $\bar{v}$, e.g. by the user specifying a desired value for $\bar{v}$, and m should be adapted to achieve the speed (note that m and $\bar{v}$ have a monotonic relationship which makes for easy adaptation).

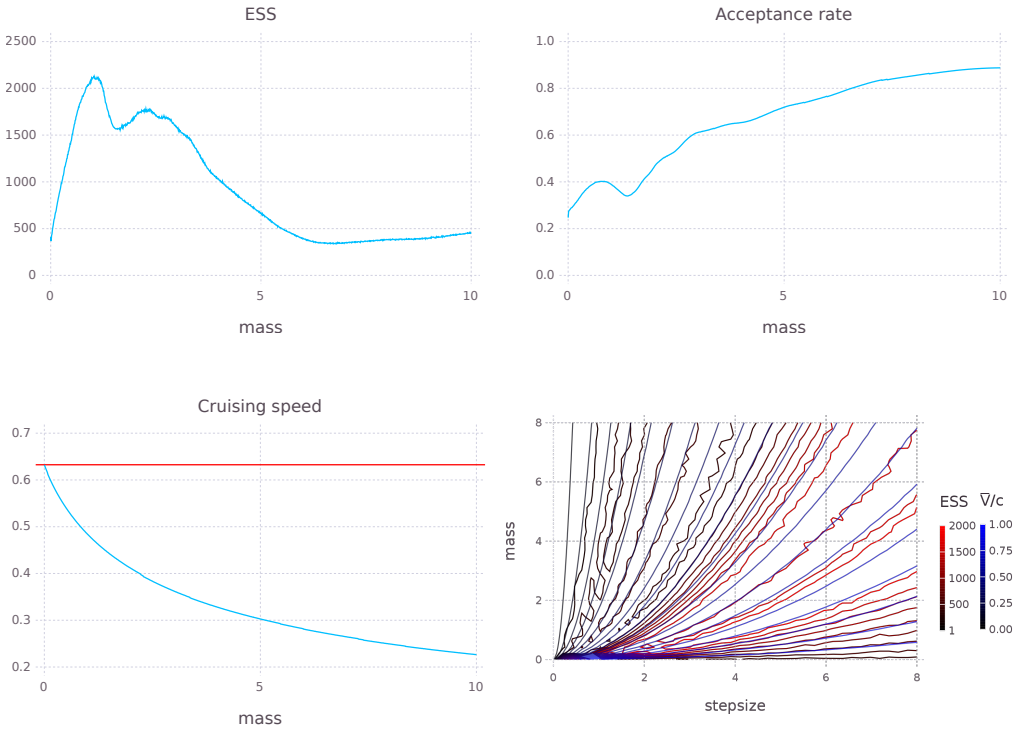


Figure 4.2: Varying m for GMM1. top row: ESS, acceptance rate; bottom row: cruising speed (the red horizontal line is c), and ESS and relative cruising speed $\bar{v}/c$ contour plots versus m and ϵ.

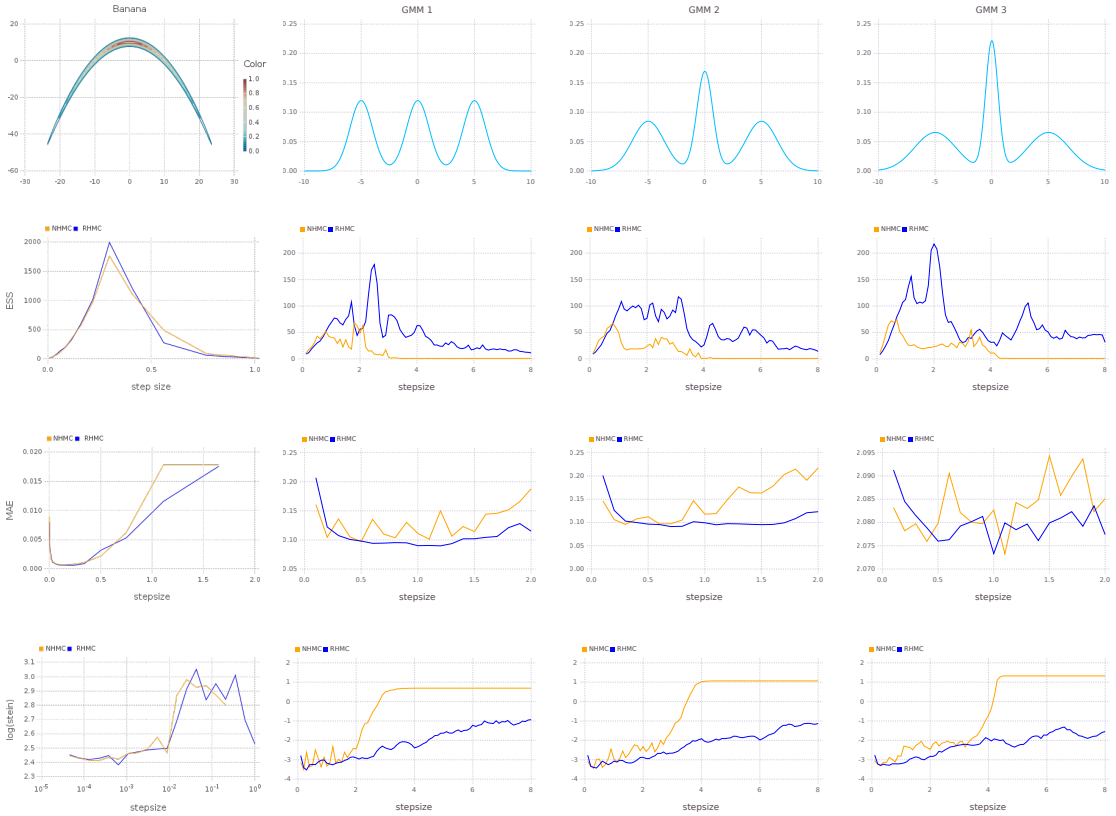


Figure 4.3: Left to right: Banana, GMM1, GMM2, GMM3 datasets. Top to bottom: density plot, ESS versus stepsize ϵ, MAE versus ϵ, log stein discrepancy versus ϵ.

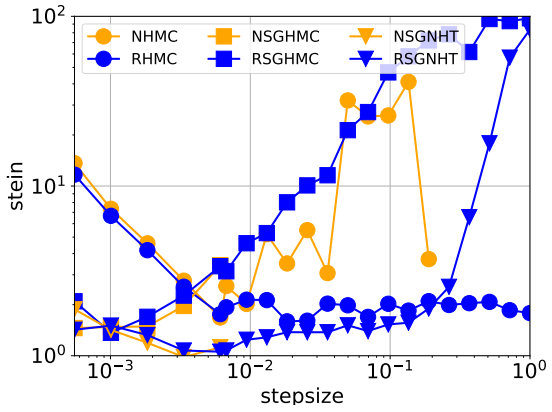


Figure 4.4: Stein discrepancy versus stepsize ϵ for logistic regression. NSGHMC and NSGNHT were unstable for $\epsilon > 6 \times 10^{-3}$ and thus their stein discrepancies were not plotted.

We next compare the performances of NHMC and RHMC for a wide range of stepsizes, via the ESS (higher better), the *mean absolute error (MAE)* between the true probabilities and the histograms of the sample frequencies (lower is better), and the log Stein discrepancy [Gorham and Mackey, 2015] which is a more accurate measure of the sample quality (lower is better). The results can be found in rows 2-4 of Figure 4.3. It can be seen that RHMC achieves better performance and is strikingly more robust to the stepsize ϵ than NHMC. As expected, this behaviour is particularly pronounced when the stepsize is large. Moreover, when the gradients of the target model span a large range of values (GMM2, GMM3), the improvements yielded by the relativistic variants are more pronounced. These results confirm that, since the speed of particles is bounded by c, RHMC is less sensitive to the presence of large gradients in the target density and more stable with respect to the choice of ϵ, allowing for a more efficient exploration of the target density.

Next we compare both the Newtonian and relativisitic variants of HMC and SGMCMC algorithms on a simulated 3-dimensional logistic regression example with 500 observations. For the stochastic versions of the algorithms, we use mini-batches of size 100. After a burn-in period of 1000 iterations, we calculated the Stein discrepancy for different ϵ while keeping the product $\epsilon \times c$ fixed. To make a fair comparison, we used 200 samples for NHMC and RHMC and 1000 samples for the SGMCMC algorithms. From Figure 4.4, we see that the relativistic variants are significantly more robust than the Newtonian variants. The NHT algorithms were able to correct for stochastic gradient noise and performed better than SGHMC algorithms. Particularly, RSGNHT had lower Stein discrepencies than other algorithms for most values of ϵ.

4.7.2 Neural Network

We then apply our algorithms to the standard MNIST dataset, which consists of 28×28 handwritten digital images from 10 classes with a training set of size $60,000$ and a test set of size $10,000$. We tested our optimization algorithm on a single layer with 100 hidden units and a multi-layer neural network with $500 * 300$ hidden units, and the results are displayed in Figure 4.5. It can be observed that our algorithm is comparable with Adam and is able to achieve a lower error rate, particularly with the 100 hidden units architecture.

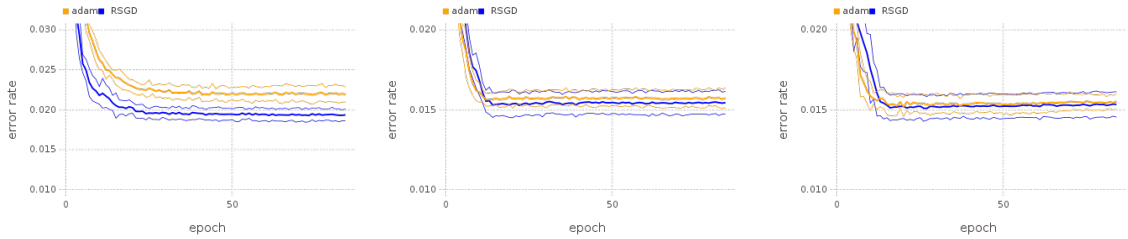


Figure 4.5: Comparison of error rate on MNIST dataset on the test set. From left to right: 100 hidden units; $500 * 300$ hidden units; $400 * 400$ hidden units.

4.8 Conclusion

In this project, we introduced a novel relativistic HMC algorithm that is based on Einstein's relativity. We also provided its stochastic variaties including RSGHMC, RSGNHT and RSGD. Moreover, we gained a good understanding on how to choose the hyper parameters c, m and ϵ. First the discretization parameter ϵ needs to be set, then we choose the maximal step $c \cdot \epsilon$ and in relation we choose the "cruising speed" $\frac{\bar{V}}{c}$ by picking m. We showed experimentally that the relativistic algorithms are much more stable, i.e., robust to the choice of parameters and noise in stochastic gradients. In particular, RSGD seems to be competitive with state-of-the-art stochastic gradient methods for fitting neural nets.

Chapter 5

On Exploration, Exploitation and Learning in Adaptive Importance Sampling

This chapter is based on the paper

- **Xiaoyu Lu**, Tom Rainforth, Yuan Zhou, Jan-Willem van de Meent and Yee Whye Teh.
  On Exploration, Exploitation and Learning in Adaptive Importance Sampling
  https://arxiv.org/abs/1810.13296
  arXiv preprint arXiv:1810.13296 (2018).

where I am the main contributor.

5.1 Abstract

We study adaptive importance sampling (AIS) as an online learning problem and argue for the importance of the trade-off between exploration and exploitation in this adaptation. Borrowing ideas from the bandits literature, we propose Daisee, a partition-based AIS algorithm. We further introduce a notion of regret for AIS and show that Daisee has $\mathcal{O}(\sqrt{T}(\log T)^{\frac{3}{4}})$ cumulative pseudo-regret, where T is the number of iterations. We then extend Daisee to adaptively learn a hierarchical partitioning of the sample space for more efficient sampling and confirm the performance of both algorithms empirically.

5.2 Introduction

Monte Carlo (MC) methods form the bedrock upon which significant sections of probabilistic machine learning and computational statistics rest. An important MC technique which forms the basis for many others is importance sampling (IS). Let $\pi(x) = f(x)/Z$ be a target density which can be evaluated pointwise up to an unknown normalising constant Z and let $q(x)$ be a proposal distribution from which samples can be drawn and can be evaluated pointwise. IS works by drawing a sequence of samples from $q(x)$ and using these to estimate Z and provide an empirical measure $\hat{p}(\cdot)$ that we can use at later time to calculate target statistics $\mathbb{E}_\pi[\phi(x)]$ for any arbitrary test function $\phi(x)$. Let $w(x_t) = f(x_t)/q(x_t)$ be the importance weight of x_t. Then,

$$Z = \int f(x)dx = \mathbb{E}_q[w(X)] \approx \frac{1}{T}\sum_{t=1}^{T} w(x_t), \tag{5.1}$$

$$\hat{p}(\cdot) = \frac{\sum_{t=1}^{T} w(x_t)\delta_{x_t}(\cdot)}{\sum_{t=1}^{T} w(x_t)}, \tag{5.2}$$

$$\mathbb{E}_\pi[\phi(x)] \approx \int \hat{p}(x)\phi(x)dx = \frac{\sum_{t=1}^{T} w(x_t)\phi(x_t)}{\sum_{t=1}^{T} w(x_t)}. \tag{5.3}$$

Note that the estimate for Z is unbiased, but that for the target statistics is biased but consistent, provided $\text{support}(f) \subset \text{support}(q)$ and $\mathbb{E}_q[w(X)\phi(X)] < \infty$ [Owen, 2013].

The efficiency of IS for a general $\phi(x)$ is governed by the choice of proposal $q(x)$, with the intuition that the closer q is to π the better. Adaptive Importance Sampling (AIS) techniques [Liu, 2008, Cappé et al., 2004, Cornuet et al., 2012, Cappé et al., 2008] attempt to improve the efficiency of IS by adapting the proposal to be closer to the target, producing a sequence of proposals $q_1, q_2, \ldots$. The IS estimates (5.3) still apply with $q(x_t)$ replaced by $q_t(x_t)$. The basic idea is that previous samples provide information about the target distribution and this information can be used to improve the proposal by altering it to better match the target [Owen, 2013, Bugallo et al., 2017].

Viewed in this way, AIS is, in effect, an online learning problem: that of learning the target density π by iteratively querying it. As opposed to the typical setup of density estimation, where each query is an iid sample from the target, here q_t is both our current estimate of the target, as well as our tool for querying the target. This exposes a trade-off between exploration and exploitation. We would like our proposal q_t to be as close as possible to the target, so that our IS estimate is as good as possible (exploit). At the same time, q_t directs where queries of the target are made and probability mass needs to be spread over the sample space where we have

high uncertainty of the target, so that we may query and reduce our uncertainty to improve our estimate in the future (explore).

In this paper we take the first steps towards developing AIS methods which optimally trade off exploration and exploitation by bringing to bear ideas from the rich literature on online learning, in particular that on multi-armed bandit algorithms [Berry and Fristedt, 1985, Auer et al., 2002, Agrawal and Goyal, 2012]. We propose the *aDAptive Important Sampling via Exploration and Exploitation* (Daisee), an algorithm which operates by partitioning the sample space into K disjoint subsets, and adapting the proposal probability of each subset by combining an exploitative estimation term and an explorative optimism boost. Based on the analysis of Chatterjee and Diaconis [2015], which showed that the Kullback-Leibler (KL) divergence from the target to the proposal is the appropriate measure of the (in)efficiency of inference using IS, we propose using the KL divergence as the measure of loss for an AIS scheme, and show that, under mild assumptions, Daisee achieves a cumulative pseudo-regret of $O(\sqrt{T}(\log T)^{\frac{3}{4}})$ where T is the number of iterations. The pseudo-regret is defined as the expectation of the regret, which is measured using the Kullback-Leibler (KL) divergence from the target to the proposal distribution. It is arguably the right measure of the efficiency of IS [Chatterjee and Diaconis, 2015]. This means that its per-iteration pseudo-regret asymptotes to zero. We also show a similar result when generalising to α-divergence [Cichocki and Amari, 2010] based regret, with $\alpha = 1$ reducing to the KL case, while $\alpha = 2$ corresponds to the variance of the importance weights $\frac{\pi(x)}{q(x)}$. Finally, we introduce and empirically investigate HiDaisee, a hierarchical extension to Daisee in which the partitions themselves are also simultaneously learned in an online fashion, leading to a more efficient overall proposal.

5.3 Related Work

The general topic of adaptive MC has received extensive attention in the literature (see e.g. [Bugallo et al., 2017, Owen, 2013] and references therein). There are a wide range of problems such as rare event simulation [Owen and Zhou, 2000, Owen et al., 2017] for which IS is more useful. Compared with MCMC, IS produces a marginal likelihood estimate. It is also the key component of many advanced MC methods.

Of particular relevance to our work, Friedman [1991], Lepage [1978] tackle the problem of multidimensional integration by recursive rectangular partitioning of the target space, where regions with higher contribution to the integral are subdivided into more subregions. Here we instead focus on the learning of the proposal densities

with exploration-exploitation trade-off in mind rather than integration. Meanwhile, He and Owen [2014] bound the regret for multiple importance sampling where the proposal distribution is a mixture of a finite number of proposal distributions, by bounding the variance of the importance sampling estimates with the help of control variates. Cappé et al. [2008] use an entropy criterion instead to learn the weights and component parameters of a mixture importance sampling density.

On the other hand, methods which address the exploration-exploitation trade off have been well-studied in online learning, most successfully under the banner of bandit algorithms [Berry and Fristedt, 1985, Bubeck et al., 2012, Srinivas et al., 2009], with Upper Confidence Bound (UCB) [Auer et al., 2002] and Thompson sampling [Agrawal and Goyal, 2012] being popular approaches. In the standard multi-armed bandit problem, one has a finite number arms and must choose an arm to pull at each iteration, returning a random reward which is distributed differently for each arm. The aim is to maximise rewards in the long run by finding the arm with the highest average reward. UCB methods operate by maintaining an estimate of the expected reward for each arm and picking arms according to the estimates plus optimism boosts which are larger for arms where our estimates are less certain to encourage exploration. Auer et al. [2002] show that UCB optimally trades-off exploration and exploitation by showing that the cumulative regret (relative to an oracle which knows the optimal arm) grows logarithmically in the number of iterations, the best growth rate achievable [Lai and Robbins, 1985]. Bubeck et al. [2013], Sen et al. [2018] extend the bandit algorithm to fat tailed distributions in absence of sub-Gaussianity.

The concept of trading off exploration versus exploitation has not been widely considered in the adaptive MC setting, though there are a number of recent works. Neufeld et al. [2014] consider the problem of choosing between a number of MC estimators, where the goal is to minimise the mean squared error by finding the estimator with lowest variance. This is still inherently a best-arm optimisation problem, rather than a true proposal adaptation. Carpentier and Munos [2011] and Leprêtre et al. [2017] consider the problem of stratified sampling for MC integration, where each stratum is viewed as an arm and the mean in each stratum is estimated. They show that in this setting the optimal approach is to choose arms in proportion to the standard deviation of the estimate produced by a single draw from the stratum. Our setting (and resultant approach) is distinct to Carpentier and Munos [2011], Leprêtre et al. [2017] as we aim to approximate a distribution, rather than performing an integration.

Algorithm 3 Daisee (aDaptive Importance Sampling with Exploration-Exploitation)

Input: a partition of the sample space $\mathcal{X}$ into K subsets $\mathcal{X}_1, \ldots, \mathcal{X}_K$, proposals for each partition $g_1(x), \ldots, g_K(x)$, unnormalized target distribution $f(x)$
Draw one sample from each subset to initialise estimates, denoting these as $x_1, \ldots, x_K$
for $t = K + 1$ **to** T **do**
    Compute proposal probabilities $\{q_{at}\}_{a=1}^K$ using (5.8), with estimates $\{\hat{Z}_{a,t-1}\}_{a=1}^K$ given by (5.7)
    Draw an arm $A_t \sim \text{Discrete}(q_{1t}, \ldots, q_{Kt})$
    Draw a sample $x_t \sim g_{A_t}(x)$
    Compute localized weight $Y_{A_t t} = f(x_t)/g_{A_t}(x_t)$ and update estimate for Z_a using (5.7)
end for

Rainforth et al. [2017] also realised the need for adaptive MC methods to trade-off between exploration and exploitation and they also take steps towards designing a framework for adapting the hierarchical partition of the space. However, their focus is on the design of a new class of meta-inference algorithms that fall outside the AIS framework and they do not consider any notion of regret nor provide a regret analysis. Our focus here is more on the exploration-exploitation trade-off in AIS, establishing a formal notion of regret so that this can be viewed as an online learning problem, and establishing a theoretical regret bound for our procedure.

5.4 Daisee

We now propose Daisee, an AIS method which has a similar flavour to UCB. Assume we have a partition of our sample space $\mathcal{X}$ into K disjoint subsets $\{\mathcal{X}_a\}_{a=1}^K$, where each subspace $\mathcal{X}_a$ can be thought as one bandit arm, and we have a fixed tractable proposal distribution $g_a(x)$ restricted on each subset $\mathcal{X}_a$ for $a \in \{1, \ldots, K\}$. For example, if $\mathcal{X}$ is compact we can choose the uniform distribution $g_a(x) = \mathbb{I}(x \in \mathcal{X}_a)/\int_{\mathcal{X}_a} dx$. Daisee uses proposal distributions of form

$$q_t(x) = \sum_{a=1}^{K} q_{at} g_a(x) \mathbb{I}(x \in \mathcal{X}_a) \tag{5.4}$$

where $\sum_a q_{at} = 1$ and the probability masses q_{at} of the subsets are to be adapted in the scheme. Note that whereas typical bandit algorithms only aim to establish the best arm, we are instead looking to asymptotically learn a distribution over how often each arm should be pulled. At each iteration t, we query the target by drawing a sample x_t from q_t. This can be achieved by first sampling an arm $A_t \sim \text{Discrete}(q_{1t}, \ldots, q_{Kt})$,

then drawing a sample $x_t \in \mathcal{X}_{A_t}$ from the subset proposal $g_{A_t}(x)$, as per lines 5 and 6 respectively in Algorithm 3.

Let the ratio $Y_{at} := \frac{f(x_t)}{g_a(x_t)}$ denote the "localized" importance weight for a sample in subset a, noting that each Y_{at}

is an unbiased estimator of the (unnormalised) target probability Z_a of $\mathcal{X}_a$:

$$\pi_a := \int_{\mathcal{X}_a} \pi(x)dx = \frac{Z_a}{Z} = \frac{Z_a}{\sum_{b=1}^K Z_b}, \tag{5.5}$$

$$Z_a := \int_{\mathcal{X}_a} f(x)dx = \mathbb{E}_{g_a}\left[\frac{f(x)}{g_a(x)}\right]. \tag{5.6}$$

At each iteration this leads to the following MC estimate for each Z_a

$$\hat{Z}_{at} := \frac{1}{N_{at}} \sum_{s \le t: A_s = a} Y_{as}, \tag{5.7}$$

where $N_{at} = \#\{s : s \le t, A_s = a\}$ is the number of times subset a was chosen up to time t. We note an unusual behaviour of the estimates $\hat{Z}_{at}$: although each Y_{at} is unbiased, each $\hat{Z}_{at}$ is biased because the N_{at}'s are random variables correlated with previous samples. We sidestep this by upper bounding the probability that $\hat{Z}_{at}$ deviates significantly, simultaneously for all t.

Naively, we can use the estimates $\hat{Z}_{at}$ to construct the next proposal, via $q_{a,t+1} \propto \hat{Z}_{at}$. The problem is that if by chance $\hat{Z}_{at}$ for some subset a is too small, the resulting underestimated proposal probability will result in low probability for the subset to be picked in future, and hence the bad estimate may not be corrected. This is a symptom of under-exploration. As in UCB, we therefore encourage exploration using an optimism boost σ_{at}:

$$q_{a,t+1} = \frac{\hat{Z}_{at} + \sigma_{at}}{\sum_{b=1}^K (\hat{Z}_{bt} + \sigma_{bt})}, \tag{5.8}$$

where σ_{at} should be decreasing with N_{at} but grow with t. The intuition, which will be formalized in the next section, is that if we have not explored the subset a sufficiently, σ_{at} is relatively large, which compensates and boosts q_{at}, allowing us to have higher chance to explore subset a and correct the under-estimate. The growth with t is to ensure sufficient exploration of all subsets over time.

As shown in Algorithm 3, Daisee iterates between drawing samples from q_t defined by (5.4), and using this sample to update our estimates (5.7), (5.8). The next section shows that this simple approach leads to a low-regret AIS strategy, in the sense that its per-round pseudo-regret tends to zero.

5.5 Regret Formalisation and Analysis

Having introduced Daisee algorithmically, we now switch focus to formalising a notion of regret suitable for the AIS setting and analysing this regret for Daisee.

In the nomenclature of bandits, Z_a can be thought of as the expected reward of arm a, while Y_{at} is the random reward received at iteration t. Note however that our aim is not to maximise expected reward, but rather to maximise the efficiency of the resulting IS sampler. Thus a measure of the (in)efficiency of the proposals q_t is required, so that we can define what makes an optimal proposal among the class in (5.4), and the regret of using a proposal relative to the optimal. We will use the KL divergence $\mathsf{KL}(\pi\|q_t)$ as this has been shown by [Chatterjee and Diaconis, 2015] to be the correct measure of the inefficiency of proposal q_t. In particular, they showed that the number of samples needed for IS to work effectively scales as $\exp(KL(\pi|q))$. For proposals as given by (5.4), this is,

$$\mathsf{KL}(\pi\|q_t) = \int_{\mathcal{X}} \pi(x) \log \frac{\pi(x)}{\sum_a q_{at} g_a(x)} dx \tag{5.9}$$

$$= \sum_a \int_{\mathcal{X}_a} \pi(x) \log \frac{\pi(x)}{g_a(x)} dx - \sum_a \pi_a \log q_{at}. \tag{5.10}$$

This loss takes an interesting form: the first term depends only on the local proposals $g_a(x)$ and the second only on the subset probability masses q_{at}. We thus see the surprising result that we can optimize separately for q_{at} and each $g_a(x)$. The closer $g_a(x)$ is to the target π on subset $\mathcal{X}_a$, the less variability there is in the weights, and the smaller the variance factor, leading to lower KL divergence. We will discuss it in more details later in section 5.7. For now, we will only consider the problem of optimising q_{at}, for which we see from (5.10) has a true optimum $q_a^* = \pi_a$ and therefore we define our regret $R(q_t)$ from using proposal q_t as

$$R_t := R(q_t) = \mathsf{KL}(\pi\|q_t) - \mathsf{KL}(\pi\|q^*) = \sum_a \pi_a \log \frac{\pi_a}{q_{at}},$$

which is just the KL divergence between the two probability vectors $(\pi_a)_{a=1}^K$ and $(q_{at})_{a=1}^K$. Given this instantaneous regret of a particular proposal, we can now use the cumulative regret $\sum_{t=1}^T R_t$ as a performance measure of the adaptation scheme itself. In particular, we would like schemes for which the regret R_t goes to zero asymptotically, which corresponds to cumulative regret that grows sublinearly.

Our regret analysis rests on an assumption that each Y_{at} is **sub-Gaussian**. Formally, there are variance factors τ_a^2, which we assume known, such that $\mathbb{E}[\exp(\theta(Y_{at} - \mathbb{E}[Y_{at}]))] \leq \exp(\frac{1}{2}\tau_a^2\theta^2)$ for each a and $\theta \in \mathbb{R}$. This might seem like a strong assumption

as importance weights are often wildly varying. However, in practice it can often be satisfied. For example, if the maximum $M = \sup_{x\in\mathcal{X}} f(x)$ can be found, and we assume $\mathcal{X}$ is compact with each $g_a(x) = \mathbb{I}(x \in \mathcal{X}_a)/\int_{\mathcal{X}_a} dx$ uniform on its subset, then τ_a can be chosen as $\frac{M}{2}\int_{\mathcal{X}_a} dx$ [Johnson-Lindenstrauss], which is $\geq \frac{Z_a}{2}$. In generally we expect τ_a of order $\mathcal{O}(Z_a)$.

Theorem 5. *Assume sub-Gaussianity and define the optimism boost as*

$$\sigma_{at} := c\tau_a\sqrt{\log t/N_{at}}$$

where $c := \sqrt{4.14\log_2(2e)}$. *The cumulative pseudo-regret at* T *of Daisee is bounded by* $\mathcal{O}(\sqrt{T}(\log T)^{\frac{3}{4}})$.

Proof. We show that the pseudo-regret $\mathbb{E}[R_t] \leq \mathbb{E}[\sum_a \pi_a \log\frac{\pi_a}{q_{at}}] = \mathcal{O}(t^{-\frac{1}{2}}(\log t)^{\frac{3}{4}})$; summing over $t = 1,\ldots,T$ gives our result. The idea is to show that $\hat{Z}_{at} \approx Z_a$ and $N_{at} \approx \mathcal{O}(t)$ with high probability, and that the pseudo-regret has the desired bound in this case. As such, define the events:

$$B_t := \{|\hat{Z}_{as} - Z_a| < \sigma_{as}\ \forall\ 1 \leq a \leq K, \frac{t}{2} \leq s \leq t\}$$
$$C_t := \{N_{at} > \beta_{at}\ \forall\ 1 \leq a \leq K\},\ \text{where}\ \beta_{at} := \frac{tZ_a}{4\sum_b(Z_b + 2c\tau_b\sqrt{\log t})}.$$

The pseudo-regret can be written as below, and the proof proceeds by bounding each term separately:

$$\mathbb{E}[R_t] = \mathbb{E}[R_t|B_t\cap C_t]\mathbb{P}(B_t\cap C_t) + \mathbb{E}[R_t|B_t^c]\mathbb{P}(B_t^c) + \mathbb{E}[R_t|B_t\cap C_t^c]\mathbb{P}(B_t\cap C_t^c). \quad (5.11)$$

For the first term in (5.11), this is just a bit of algebra:

$$\begin{aligned}
\mathbb{E}[R_t|B_t\cap C_t]\mathbb{P}(B_t\cap C_t) \leq& \mathbb{E}[\sum_a \pi_a \log\frac{\pi_a}{q_{at}}|B_t\cap C_t] \\
\leq& \mathbb{E}[\sum_a \pi_a \log(\frac{\pi_a}{Z_a}\sum_b(Z_b + 2\sigma_{bt}))|B_t\cap C_t] \\
=& \mathbb{E}[\log(1 + \frac{2}{Z}\sum_a \sigma_{at})|B_t\cap C_t] \\
\leq& \frac{2}{Z}\sum_a \mathbb{E}[\sigma_{at}|B_t\cap C_t] \\
\leq& \frac{2}{Z}\sum_a c\tau_a(\log t)^{\frac{1}{2}}\beta_{at}^{-\frac{1}{2}} \\
=& \mathcal{O}(t^{-\frac{1}{2}}(\log t)^{\frac{3}{4}}\frac{\sqrt{\sum_a \tau_a}}{\sum_a Z_a}\sum_a \frac{\tau_a}{\sqrt{Z_a}}). \qquad (5.12)
\end{aligned}$$

For the second and third terms in (5.11), the main steps are to show that both $\mathbb{P}(B_t^c)$ and $\mathbb{P}(C_t^c|B_t)$ are small. For $\mathbb{P}(B_t^c)$, for each a, we can rewrite $\hat{Z}_{as}$ as $\hat{Z}_{as} = \frac{1}{N_{as}} \sum_{i=1}^{N_{as}} Y_{at_i^a}$, where we re-index the time $\{t_i^a\}_{i=1}^{N_{as}} := \{j : A_j = a, 1 \leq j \leq s\}$. Recall that each $Y_{at_i^a}$ is sub-Gaussian with mean Z_a and variance factor τ_a^2. The complication here is that there is a non-trivial dependence of these and the N_{as}'s through the proposal probabilities q_{as}'s. To side-step this dependence, we can use a finite time law of the iterated logarithm [Balsubramani, 2014, Koolen, 2017] (see theorem 6).

Theorem 6. *(Finite Time Law of the Iterated Logarithm [Balsubramani, 2014, Koolen, 2017]) Let $X_1, X_2, \ldots$ be a sequence of iid sub-Gaussian random variables with mean μ and variance factor τ^2. For each $\delta \in (0,1)$, we have*

$$\mathbb{P}\left(\exists s \geq 1, \left|\frac{1}{s}\sum_{i=1}^{s} X_i - \mu\right| \geq \sqrt{\frac{2.07\tau^2}{s}\log\left(\frac{2}{\delta}(1+\log_2 s)^2\right)}\right) \leq \delta. \tag{5.13}$$

We apply theorem 6 to bound the deviations of $\hat{Z}_{as}$, for all possible values of N_{as} between 1 and s. Choosing $\delta = 2s^{-2}$, we have

$$\sqrt{\frac{2.07\tau_a^2}{N_{as}}\log\left(\frac{2}{\delta}(1+\log_2 N_{as})^2\right)} \leq \sqrt{\frac{2.07\tau_a^2}{N_{as}}\left(2\log s + 2\frac{\log s}{\log 2}\right)} = \sigma_{as} \tag{5.14}$$

Hence,

$$\begin{aligned}
&\mathbb{P}(|\hat{Z}_{as} - Z_a| > \sigma_{as}) && (5.15)\\
&\leq \mathbb{P}\left(\exists l, 1 \leq l \leq s : \left|\frac{1}{l}\sum_{i=1}^{l} Y_{at_i^a} - Z_a\right| > \sqrt{\frac{2.07\tau_a^2}{l}\log\left(\frac{2}{\delta}(1+\log_2 l)^2\right)}\right)\\
&\leq \delta = 2s^{-2} && (5.16)
\end{aligned}$$

Using a union bound,

$$\mathbb{P}(B_t^c) \leq \sum_{a=1}^{K}\sum_{s=[\frac{t}{2}]}^{t} \mathbb{P}(|\hat{Z}_{as} - Z_a| > \sigma_{as}) \leq \sum_{a=1}^{K}\sum_{s=[\frac{t}{2}]}^{t} 2s^{-2} = \mathcal{O}(Kt^{-1}). \tag{5.17}$$

To bound $\mathbb{P}(C_t^c|B_t)$, note that conditioned on B_t we have, for $\frac{t}{2} \leq s \leq t$ and each a:

$$\sum_{s=1}^{t} q_{as} \geq \sum_{s=\frac{t}{2}}^{t} q_{as} \geq \sum_{s=\frac{t}{2}}^{t} \frac{Z_a}{\sum_b Z_b + 2\sigma_{bs}} \geq \frac{Z_a t}{2\sum_b (Z_b + 2c\tau_b\sqrt{\log t})} = 2\beta_{at}. \tag{5.18}$$

Theorem 7. *(Azuma–Hoeffding Inequality [Azuma, 1967]). Let $D_1, D_2, \ldots$ be a martingale difference sequence, and suppose that $|D_k| \leq b_k$ almost surely for all $k \geq 1$. Then for all $n \geq 0, \mathbb{P}(|\sum_{i=1}^n D_i| \geq \epsilon) \leq 2\exp\left(-\frac{\epsilon^2}{2\sum_{i=1}^n b_i^2}\right)$.*

We can write $N_{at} = \sum_{s=1}^t W_{as}$ where $W_{as}|\mathcal{F}_s \sim$ Bernoulli(q_{as}) and $\mathcal{F}_s$ is the information filtration up to iteration s. Since $(W_{as} - q_{as})_{s=1}^{\infty}$ is a martingale difference sequence, the Azuma-Hoeffding Inequality [Azuma, 1967] (see theorem 7) gives:

$$\mathbb{P}(N_{at} < \beta_{at}|B_t) \leq \mathbb{P}(N_{at} < \frac{1}{2}\sum_{s=1}^t q_{as}|B_t) \leq 2\exp\left(-\frac{1}{2t}\beta_{at}^2\right). \tag{5.19}$$

A union bound along with $e^{-\alpha} < 1/\alpha$ then shows that $\mathbb{P}(C_t^c|B_t) \leq \mathcal{O}(Kt^{-1}\log t)$. Putting everything together, along with a loose bound of $\mathbb{E}[R_t|B_t^c], \mathbb{E}[R_t|C_t^c \cap B_t] \leq \mathcal{O}(\log t)$ (see Appendix 5.9.1), gives the result that $\mathbb{E}[R_t]$ is dominated by the first term, which is $\mathcal{O}(t^{-\frac{1}{2}}(\log t)^{\frac{3}{4}})$. □

It is worth elaborating on the dependence of (5.12) on τ_a's and Z_a's. Recall that in general we expect $\tau_a = \mathcal{O}(Z_a)$. Suppose $\tau_a \approx \Omega Z_a$ for some $\Omega > 0$. Then $\frac{(\sum_a \tau_a)^{1/2}}{\sum_a Z_a}\sum_a \frac{\tau_a}{Z_a^{1/2}} \approx \Omega^{3/2}\frac{\sum_a Z_a^{1/2}}{(\sum_a Z_a)^{1/2}} \leq \Omega^{3/2}K^{1/2}$. The larger the variance factors (larger Ω), and the larger the number of arms K, the worse the bound. Another harder scenario occurs when masses are not uniformly distributed across the arms, and we have no a priori knowledge of which arm has high mass. That is, suppose $Z_{\max} = Z_1 \gg Z_a = Z_{\min} > 0$ for $a > 1$, and $\tau_a \approx \Omega Z_{\max}$ for all a. Then $\frac{(\sum_a \tau_a)^{1/2}}{\sum_a Z_a}\sum_a \frac{\tau_a}{Z_a^{1/2}} \approx \Omega^{3/2}K^{3/2}(\frac{Z_{\max}}{Z_{\min}})^{\frac{1}{2}}$, so the difficulty of the problem is governed by the square root of the ratio $Z_{\max}/Z_{\min}$, and there is an extra factor of K due to the higher regret of searching K arms only to find just one of them has high mass.

In Appendix 5.9.2 we generalise the result to regret defined by α-divergence losses, obtaining a cumulative pseudo-regret bound of $\mathcal{O}(t^{-\frac{1}{2}}(\log t)^{\frac{1}{2}(\frac{1}{2\alpha}+1)})$. As $\alpha \to 1$, the α-divergence reduces to the KL divergence and the regret bounds coincide. When $\alpha = 2$ and Z is known, the loss is $\frac{1}{2}\int_{\mathcal{X}}(\frac{\pi(x)}{q_t(x)} - 1)^2 q_t(x)dx$, which is the variance of the importance weights.

5.6 Experiments

5.6.1 Daisee on 1D Example

We first demonstrate Daisee on a simple problem, empirically evaluating a number of forms of the optimism boost. Our sample space is the unit interval $[0, 1]$ and we partitioned it evenly into 100 subintervals. Our target density $\pi(x)$ and the

subproposals $g_a(x)$ were such that Y_{at} has distribution $2a\text{Bernoulli}(1/100)/101$ for interval $a \in \{1, \ldots, 100\}$. We made this choice so that Y_{at} has a large but controllable variance, so that the resulting problem of adaptation is hard enough for exploration to be important. τ_a are selected to be proportional to the volume of the subspace (same for all arms), and is tuned to minimise regret, after which we fixed its value and repeat the experiments. We did not find the regret very sensitive to the values of τ_a. Figure 5.1b shows that our algorithm successfully recovers an effective proposal. Figure 5.1c compares the cumulative regret for various possible optimism boosts, each of which involve an (not shown) constant multiplier that has been optimised as a hyperparameter. We observe that optimism boosts with an inverse relationship with N_{at} and slow growth with t work well, and seem to achieve sublinear cumulative regret. Among different forms of the boost, it can be observed that our defined optimism boost gives the lowest cumulative regret.

5.6.2 Validation of Theorem 5

In this section, we empirically evaluate the dependency of regret on different parameters, including the number of arms K, the variance factor τ and the ratio Z_{max}/Z_{min} where $Z_{max} = \max_a\{Z_a\}_{a=1}^K$ and $Z_{min} = \min_a\{Z_a\}_{a=1}^K$. Instead of tuning τ by grid search as in 5.6.1, we assume it is known and we design the target density accordingly. We adapt the target density such that when varying each parameter, the other two are kept fixed. The sample space we consider is the interval $(0, 1)$ and we partition the space into equi-spaced intervals.

Regret bound vs τ: We fix the number of arms $K = 10$. The unnormalised density $f(x) = (10 + \delta)\mathbb{1}_{(0<x\leq 0.05)} + (10 - \delta)\mathbb{1}_{(0.05<x\leq 0.1)} + 0.1\mathbb{1}_{(0.1<x<1)}$, for δ ranging from 0.001 to 8 which results in a range of different variance factor τ while keeping K and the ratio Z_{max}/Z_{min} unchanged. It can be seen from Figure 5.2a that bigger τ leads to larger regret which coincides with our theorem. Intuitively, bigger τ indicated larger fluctuations which is harder and leads to larger regret.

Regret bound vs K: We set $f(x) = 3K\mathbb{1}_{(0<x\leq 0.2)} + K\mathbb{1}_{(0.2<x<1)}$ and vary K from 5 to 100. It can observed from Figure 5.2b that larger K leads to larger regret, as derived from the theorem. This may seem confusing at first, but notice that we defined the regret to be the second term of the KL in (5.10). What it means is that we are considering the regret compared to the optimal proposal in each of class with different K: it is easier to estimate relative masses over a smaller number of subspace and thus smaller regret for smaller K. However, the overall KL decreases when K increases.

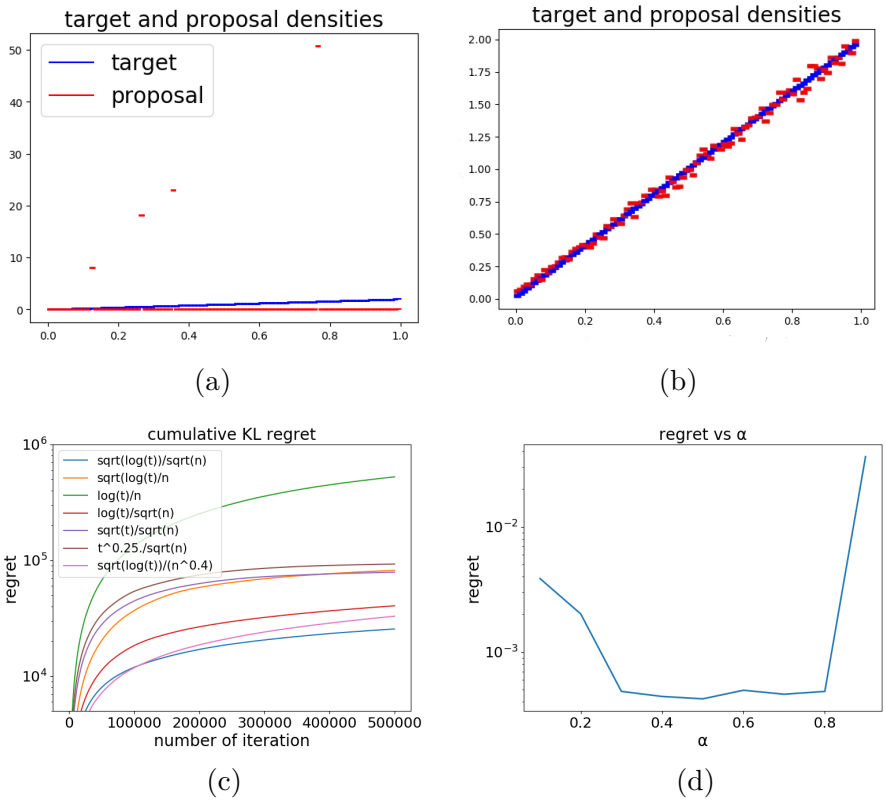
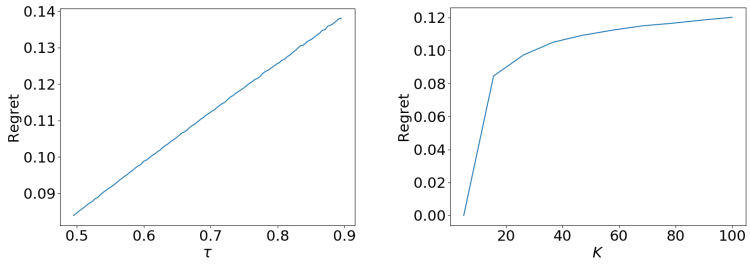


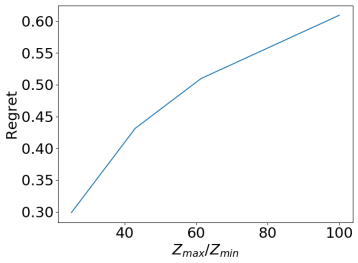
Figure 5.1: Results for 1d example. (a)(b) Target and proposal probabilities at final iteration, with no optimism boost and with optimism boost $\sqrt{\log(t)/N_{at}}$ respectively; (c) Cumulative regret as functions of iteration, averaged over 10 runs for different forms of optimism of boost (σ_{at}). We observe that our chosen σ_{at} outperforms the others; (d) We explore the amount of exploration more systematically by considering boost of the form $\sigma_{at} = (\log(t)/N_{at})^{\alpha}$ and reporting the final instantaneous regret as a function of α, averaged over 10 runs. It can be observed that values of α near to 0.5 gave the lowest regret, matching the theory.

Regret bound vs Z_{max}/Z_{min}: In this case, we move some mass in the first arm equally to the rest of the arms. $f(x) = 10\mathbb{1}_{(0<x\leq 1/K-\delta)} + 0.1\mathbb{1}_{(1/K-\delta<x\leq 1/K)} + 10\mathbb{1}_{(float(10x)<\delta/(K-1))} + 0.1\mathbb{1}_{(float(10x)\geq\delta/(K-1))}$ where $float$ indicates the floating part of the number. Again, the result from 5.2c shows that the result is as expected according to the theorem. The bigger the ratio, the more variations there is in the target distributions, and more samples needs to be drawn to account for the uncertainties.

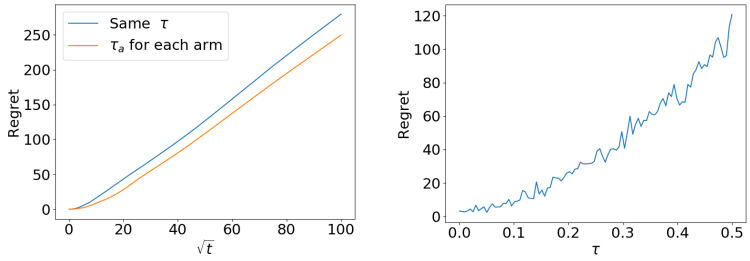
Using separate τ_a on each arm: If one knows each τ_a on arm a, then the regret should have a lower bound according to our derivation in (5.12). We illustrate this on a simple problem where the target unnormalised density is $f(x) = 20\mathbb{1}_{0<x\leq 0.25} + 3\mathbb{1}_{0.25<x\leq 0.5} + 9\mathbb{1}_{0.5<x<\leq 0.99} + \mathbb{1}_{0.99<x<1}$ and we set $K = 5$. The result can be found in figure 5.2d where with different τ_a according to the distribution over each arm leads



(a) (b)



(c)

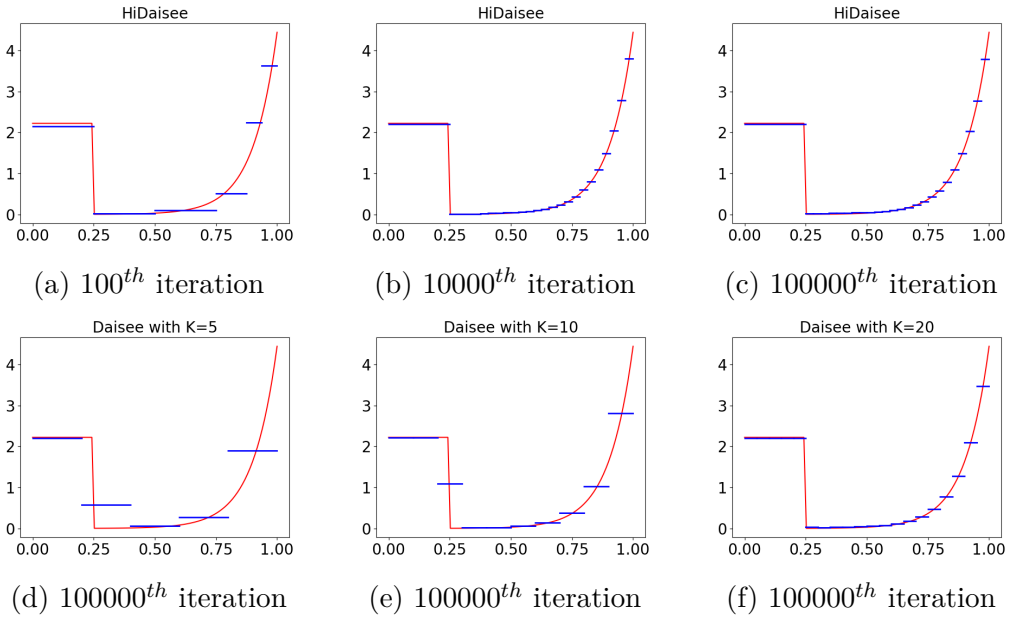


(d) (e)

Figure 5.2: (a)-(c) Instantaneous regret versus τ, K and Z_{max}/Z_{min} respectively, while keeping the other parameters fixed. (d) Cumulative regret versus $\sqrt{t}$ when using the same τ or different τ_a for each arm. (e) Sensitivity analysis of cumulative regret on τ for a single target density.

to better performance. It can also be seen that the cumulative regret grows roughly linearly with $\sqrt{t}$, meaning our analysis for the regret is quite tight.

Sensitivity analysis on τ: In practice, τ may not be estimated accurately and thus requires tuning. We tuned τ in the same way as in section 5.6.1 where it is chosen to minimise the regret, and show that the algorithm is robust for small values of τ. With the same set up as in figure 5.2d, we use the same τ for all arms and is tuned via grid search with 10 repetitions, the resulting regret displayed in figure 5.2e verifies that Daisee is not very sensitive to τ.



(a) 100^{th} iteration (b) 10000^{th} iteration (c) 100000^{th} iteration

(d) 100000^{th} iteration (e) 100000^{th} iteration (f) 100000^{th} iteration

Figure 5.3: Results using HiDaisee. Target densities are plotted in red and the adaptive proposal probabilities are plotted in blue. (a)-(c): results for HiDaisee at iteration 1000, 10000 and 100000 respectively. It can be seen that HiDaisee stops splitting the region where the density is relatively high but flat. (d)-(f): results for Daisee with difference number of partitions $K = 5, 10, 20$ respectively, at iteration 100000, where the space has been partitioned into K intervals of equal length.

5.7 Hierarchical Daisee

With a view to improving the efficiency of Daisee, we now extend it by introducing an approach to refining partition proposals g_a in part of the space where g_a does not approximate f well. Since we have a disjoint set of proposals g_a on each arm $\mathcal{X}_a$, one approach of adapting the g_a is to seek a better partitioning over the space. A good partitioning should be finely grained in areas where the density is highly fluctuating, and we consider adapting the partition alongside the subset probabilities q_{at} using information from previous the samples. So, whereas Daisee used a set of fixed partitions, we will now adapt the partition by recursively splitting subsets into two halves. We refer to this extended approach as *HiDaisee* (Hierarchical Daisee). We now describe algorithmic details and show some promising experimental results for HiDaisee, but leave the theoretical analysis to future work.

For simplicity, we consider a finite rectangle for $\mathcal{X}$, and a hierarchical binary partition of $\mathcal{X}$ whereby each split simply splits a rectangle into two equal rectangles along one dimension (we simply cycle over the dimensions in the experiments). At each point of the algorithm the partition consists of rectangular subsets, and we use

Algorithm 4 HiDaisee (Hierarchical Adaptive Importance Sampling with Exploration-Exploitation)

Input: proposal distribution $g(x)$, unnormalized target distribution $f(x)$, ESS threshold $\alpha \in (0, 1)$, minimum number of samples for splitting node $N_{\min}$, tree initialisation $\mathcal{T}$
for $t = 1$ **to** T **do**
    Set node id to root $i \leftarrow 0$ and initialise traversal path to be empty $\mathcal{P} \leftarrow \emptyset$
    $i \notin$ leaf nodes of $\mathcal{T}$
    $\mathcal{P} = \mathcal{P} \cup i$
    $r \sim \text{Uniform}(0, 1)$
    $q_{\text{left}} = \frac{q_{i_\ell}}{q_{i_\ell} + q_{i_r}}$ i_ℓ and i_r indicate left and right children of i
    if $r < q_{\text{left}}$ **then** $i \leftarrow i_\ell$ **else** $i \leftarrow i_r$ **end if**
    Draw $x_t \sim g_i(x)$ and compute weight $Y_{it} = f(x_t)/g_i(x_t)$
    Update q_j for all $j \in \mathcal{P}$ ▷ Update q_i for leaf as per (5.8) and all ancestors as per (5.21)
    Update ESS_i and N_i for the leaf node
    if $N_i \geq N_{\min}$ and $\text{ESS}_i < \alpha N_i$ **then**
        Split the node, updating $\mathcal{T}$ to include new nodes and calculating the corresponding q_j by pushing samples down tree
    end if
end for

uniform subproposals $g_a(x)$. In order to learn the tree in an online fashion, HiDaisee decides whether to split a leaf node or not whenever it is sampled. We want to avoid oversplitting the tree as larger trees are more expensive to sample from and require more memory. To control this, we introduce a *splitting criterion*, such that whenever a node is chosen, it is split if it passes the splitting criterion. We want to stop partitioning the subspace if the importance weights obtained from that subspace are similar to each other, as this indicates little is to be gained by further splitting. To this end we introduce a splitting criterion based on the *effective Sample Size* (ESS) [Owen, 2013] of the node, a measure of efficiency in importance sampling that indicates the quality of the proposal. For rectangle $\mathcal{X}_a$ at iteration t, the $\text{ESS}_{at} \in [1, N_{at}]$ is defined as

$$\text{ESS}_{at} := \frac{(\sum_{l \leq t: A_l = a} Y_{al})^2}{\sum_{l \leq t: A_l = a} Y_{al}^2} \tag{5.20}$$

where Y_{al} are the local importance weights as before. Our split criterion is then to only split when both a) the number of samples at the node exceeds a certain threshold $N_{\min}$ (set to 10 in our experiments) and b) the ESS of the samples is less that αN_{at}, where α is an ESS threshold parameter.

Though the leaves of the tree form a valid partitioning for Daisee and can thus be sampled directly, the computational complexity of doing so naïvely scales as the number of subsets K and so becomes inefficient as the the tree becomes large. By storing running estimates for each node i in the tree,

$$q_{it} = \sum_{a \in \text{leaves}(i)} q_{at}, \tag{5.21}$$

where the sum is over the leaves under node i, we can instead traverse down the tree by recursively choosing the left or right child node until a leaf node is reached. The computational cost now scales with the depth of the tree ($\approx \mathcal{O}(\log K)$) rather than K. Putting everything together, we arrive at the complete HiDaisee algorithm as shown in Algorithm 4, where we will sometimes omit an implicit dependency on t to avoid clutter.

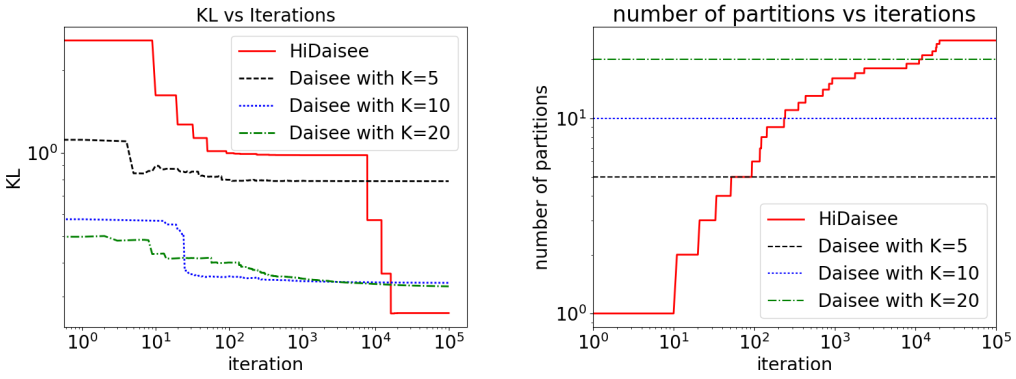
5.7.1 Experiment Results for HiDaisee

We first demonstrate HiDaisee on a simple example where the target density is $\pi(x) \propto \exp(10(x-1))\mathbf{1}_{x\in(0.25,1)} + 0.5\mathbf{1}_{x\in(0,0.25]}$. Figure 5.3 shows the learned proposals at the different numbers of iterations, comparing HiDaisee to Daisee. It can be seen that initially HiDaisee overestimates the region with low density to encourage exploration. The algorithm stops splitting the region on $0 < x < 0.25$ where the density is relatively high but is flat, whereas it continues to split for the rightmost region where the density is high and is not flat. A video of the learning evolution is available at https://www.youtube.com/watch?v=LG5RCBcs4kg.

Figure 5.4a shows a comparison of the KL loss per iteration for HiDaisee and Daisee for this example. Though the initial performance of HiDaisee is worse than Daisee, because it initially has $K = 1$, it quickly catches up and has the best final KL loss. It takes longer for HiDaisee to converge as it is a harder problem when starting with $K = 1$. We also plot the evolution of the number of partitions in Figure 5.4b. Here it can be seen that the algorithm converges to a fixed partition, which is the optimal balancing between accuracy and time complexity.

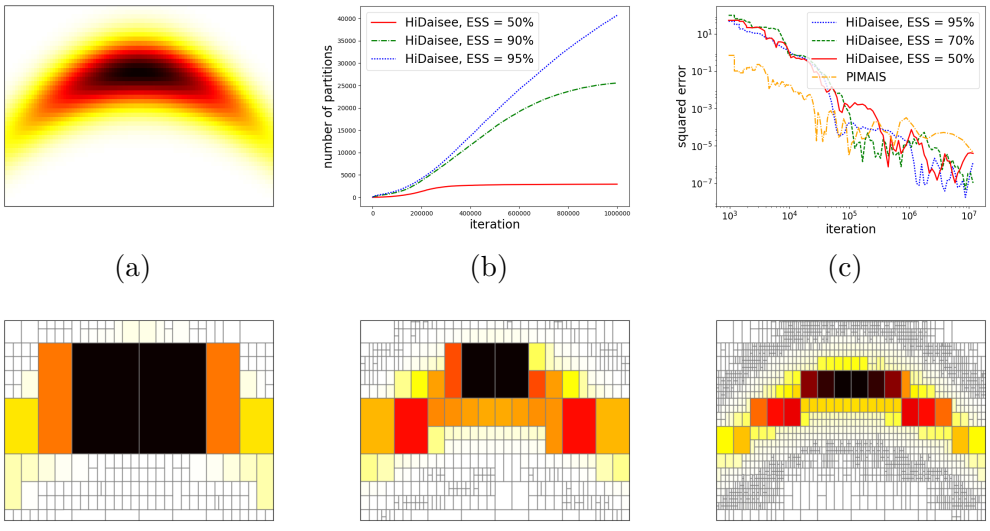
We also applied HiDaisee to the classic $2D$ banana shaped problem, with density $f(x_1, x_2) \propto \exp\{-0.5(0.03x_1^2 + (x_2 + 0.03(x_1^2 - 100))^2)\}$. The results are displayed in Figure 5.5. We see that the region with high density stops splitting when the density is relatively flat with our ESS splitting criterion. We compare HiDaisee with *Parallel Interacting Markov Adaptive Importance Sampling* (PI-MAIS) [Martino et al., 2017] for calculating the marginal likelihood (the normalising constant Z) in Figure 5.5c,

which shows that when there are more samples and more complex hierarchies are constructed, HiDaisee achieves lower squared error.



(a) (b)

Figure 5.4: Comparison of Daisee and HiDaisee. (a):$KL(\pi||q_t)$ vs iteration. Conclusion: Daisee with fixed partition converges to the best proposal within that class, which can still be far from the true target density. For HiDaisee the KL continues to decrease; (b): number of partitions vs iterations, which grows sub linearly with our splitting criterion.



(a) (b) (c)

(d) $ESS = 50\%$ (e) $ESS = 70\%$ (f) $ESS = 95\%$

Figure 5.5: Banana shaped example, (a):target density; (b): number of partitions versus number of iterations for different ESS as a percentage of number of samples ; (c): Comparision of HiDaisee and PI-MAIS in estimating the marginal likelihood $(\hat{Z} - Z)^2$, results are averaged over 10 runs; (d)-(f) learned proposal distributions with difference ESS, showing the partitions.

5.8 Conclusions

In this work, we have addressed the issue of exploration-exploitation in adaptive importance sampling, and proposed a novel approach through the lens of multi-armed bandit problems, borrowing the ideas of upper confidence bounds. We show a cumulative pseudo-regret of $\mathcal{O}(\sqrt{T}(\log T)^{\frac{3}{4}})$. We extend our method to the hierarchical case, where the sample space is recursively split in high density regions, and demonstrate experimentally that our methods give promising performance with little computational costs. Future work includes the relaxation of the sub-Gaussian assumption , an investigation of lower bounds on the KL regret in the AIS setting, and the theoretical analysis of (various variants of) HiDaisee. The analysis already introduced is an important step towards the analysis for HiDaisee, which requires substantial complications and is left as future work. It would also be interesting to investigate the application of the ideas we introduced to other adaptive MC methods.

5.9 Appendix

5.9.1 Proof Details of Theorem 5

Note that for any event E, by Jensen's inequality,

$$\mathbb{E}[R_t|E] \leq \sum_a \pi_a \log\left(\pi_a \mathbb{E}\left[\frac{\sum_b(\hat{Z}_{bt}+\sigma_{bt})}{\hat{Z}_{at}+\sigma_{at}}|E\right]\right). \tag{5.22}$$

Bounding $\mathbb{E}[R_t|C_t^c \cap B_t]$:

$$\begin{aligned}
\mathbb{E}[R_t|C_t^c \cap B_t] &\leq \sum_a \pi_a \log\left(\pi_a \mathbb{E}\left[\frac{\sum_b(Z_b+2\sigma_{bt})}{Z_a}|C_t^c \cap B_t\right]\right) && (5.23)\\
&\leq \frac{2}{Z}\sum_a \mathbb{E}[\sigma_{at}|C_t^c \cap B_t] && (5.24)\\
&= \frac{2}{Z}\sum_a \mathbb{E}\left[c\tau_a\sqrt{\frac{\log t}{N_{at}}}|C_t^c\right] && (5.25)\\
&\leq \frac{2c\sum_a \tau_a}{Z}\sqrt{\log t}. && (5.26)
\end{aligned}$$

Bounding $\mathbb{E}[R_t|B_t^c]$: recall (5.22), we first bound

$$\mathbb{E}\left[\frac{\sum_b(\hat{Z}_{bt}+\sigma_{bt})}{\hat{Z}_{at}+\sigma_{at}}|B_t^c\right] \leq \mathbb{E}\left[\frac{\sum_b(\hat{Z}_{bt}+\sigma_{bt})}{\sigma_{at}}|B_t^c\right] \leq \frac{\sqrt{t}}{c\tau_a\sqrt{\log t}}\sum_b \mathbb{E}[(\hat{Z}_{bt}+\sigma_{bt})|B_t^c]. \tag{5.27}$$

Now

$$\mathbb{E}[(\hat{Z}_{bt}+\sigma_{bt})|B_t^c] \leq \mathbb{E}[(\hat{Z}_{bt}+\sigma_{bt})|B_t^c, |\hat{Z}_{bt}-Z_b|>\sigma_{bt}] + \mathbb{E}[(\hat{Z}_{bt}+\sigma_{bt})|B_t^c, |\hat{Z}_{bt}-Z_b|\leq\sigma_{bt}]$$

where the second term is bounded by $Z_b + 2c\tau_b\sqrt{\log t}$. For the first term, we use properties of sub-Gaussian distributions:

$$\mathbb{E}[(\hat{Z}_{bt}+\sigma_{bt})|B_t^c, |\hat{Z}_{bt}-Z_b|>\sigma_{bt}] \leq \sum_{n=1}^{t}\mathbb{E}[\hat{Z}_{bt}+\sigma_{bt}||\hat{Z}_{bt}-Z_b|>\sigma_{bt}, N_{bt}=n] \quad (5.28)$$

$$= \sum_{n=1}^{t}\mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n}(Y_{bt_j}^b - Z_b) + Z_b + c\tau_b\frac{\sqrt{\log t}}{n} | \frac{1}{n}\sum_{j=1}^{n}(Y_{bt_j}^b - Z_b) > c\tau_b\frac{\sqrt{\log t}}{n}\right] \quad (5.29)$$

$$\leq \sum_{n=1}^{t}\mathbb{E}[W_n|W_n > c\tau_b\sqrt{\log t}] + t(Z_b + c\tau_b\sqrt{\log t}), \quad (5.30)$$

where $W_n := \frac{1}{n}\sum_{j=1}^{n}(Y_{bt_j}^b - Z_b)$ is also sub Gaussian distributed with zero mean and variance proxy $\frac{\tau_b^2}{n}$ since it is a linear combination of n sub Gaussian distributed random variables. For large t the mean of the truncated sub Gaussian $\mathbb{E}[W_n|W_n > c\tau_b\sqrt{\log t}]$ is upper bounded by that of a truncated Gaussian:

$$\mathbb{E}[W_n|W_n > c\tau_b\sqrt{\log t}] \leq \tau\frac{\phi(c\tau_b\sqrt{\log t})}{1-\Phi(c\tau_b\sqrt{\log t})} = \frac{2\phi(c\tau_b\sqrt{\log t})}{\mathrm{erfc}(\frac{c\tau_b\sqrt{\log t}}{\sqrt{2}})} \quad (5.31)$$

$$\leq \frac{\phi(c\tau_b\sqrt{\log t})}{\exp(-\frac{c^2\tau_b^2\log t}{2})}\sqrt{\pi}\left(\frac{c\tau_b\sqrt{\log t}}{\sqrt{2}} + \sqrt{\frac{c^2\tau_b^2\log t}{2}+2}\right) \quad (5.32)$$

$$\leq \sqrt{2}c\tau_b\sqrt{\log t}, \quad (5.33)$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the *pdf* and *cdf* of a standard normal distributions and erfc denotes the complementary error function, we used [Kschischang, 2017] for the inequality involving erfc above. Hence we have $\mathbb{E}[(\hat{Z}_{bt}+\sigma_{bt})|B_t^c] \leq \mathcal{O}(t\sqrt{\log t})$ and combinaing (5.22)(5.27)(5.30)(5.33) gives

$$\mathbb{E}[R_t|B_t^c] \leq \sum_a \pi_a \log(\frac{\pi_a}{c}\sqrt{\frac{t}{\log t}}\sum_b(\sqrt{2}c\tau_b t\sqrt{\log t} + Z_b) \leq \mathcal{O}(\log t). \quad (5.34)$$

5.9.2 Alpha Loss Family

We can generalise the KL loss to the family of alpha-loss which is associated with α -divergence for $\alpha \in \mathbb{R}$ [Cichocki and Amari, 2010]. For any target (unnormalised)

density p and proposal $\sim q$, define:

$$\mathcal{L}_\alpha(x,q) = \frac{1}{\alpha(1-\alpha)}\left(\alpha\frac{\pi(x)}{q(x)} + (1-\alpha) - \left(\frac{\pi(x)}{q(x)}\right)^\alpha\right)$$

where we take the limit for $\alpha = 0, 1$. Taking expectation with respect to x yields the $\alpha-$divergence between p and q:

$$\mathcal{L}_\alpha(q) = \mathbf{E}[\mathcal{L}_\alpha(x,q)] = D_\alpha(\pi||q) = \frac{1}{1-\alpha}Z + \frac{1}{\alpha} - \frac{1}{\alpha(1-\alpha)}\int_{\mathcal{X}}\left(\frac{\pi(x)}{q(x)}\right)^\alpha q(x)dx$$

In particular, $\mathcal{L}_1(q) = KL(\pi||q), \mathcal{L}_0(q) = KL(q||\pi)$ and when $\alpha = 2$, this leads to the $\mathcal{L}_2$ loss: $\mathcal{L}_2(q) = D_2(p||q) = \frac{1}{2}\int_{\mathcal{X}}\left(\frac{\pi(x)}{q(x)} - 1\right)^2 q(x)dx$. Using Lagrangian gives the optimal proposal

$$q_a^* = \frac{\left(\int_{\mathcal{X}_a}\pi(x)^\alpha g_a(x)^{1-\alpha}dx\right)^{\frac{1}{\alpha}}}{\sum_a\left(\int_{\mathcal{X}_a}\pi(x)^\alpha g_a(x)^{1-\alpha}dx\right)^{\frac{1}{\alpha}}} := \frac{(\pi_a^{(\alpha)})^{\frac{1}{\alpha}}}{\sum_a(\pi_a^{(\alpha)})^{\frac{1}{\alpha}}} \tag{5.35}$$

where $\pi_a^{(\alpha)} := \int_{\mathcal{X}_a}\pi(x)^\alpha g_a(x)^{1-\alpha}dx$. The optimal loss is therefore $\mathcal{L}_\alpha(q^*) = \frac{1}{1-\alpha} + \frac{1}{\alpha} + \frac{1}{\alpha(\alpha-1)}(\sum_a \pi_a^{(\alpha)})^\alpha$. Similarly as in section 5.4, we can use a Monte Carlo estimate for the intractable optimal q^*:

$$q_a^* \propto \pi_a^{(\alpha)} = \frac{1}{Z}\int_{\mathcal{X}_a}\left(\frac{f(x)}{g_a(x)}\right)^\alpha g_a(x)dx \propto \mathbb{E}_{g_a}\left[\left(\frac{f(x)}{g_a(x)}\right)^\alpha\right]. \tag{5.36}$$

Replacing the definition of Y_{al} with $Y_{al} := \left(\frac{f(x_l)}{g_a(x_l)}\right)^\alpha$, and refine $\hat{Z}_{a,t+1}$ as in (5.7), *i.e.* $\hat{Z}_{a,t+1} := \frac{\sum_{l\le t:A_l=a} Y_{al}}{N_{a,t+1}}$, according to (5.35) we propose

$$q_{a,t+1} = \frac{(\hat{Z}_{a,t+1} + \sigma_{a,t+1})^{\frac{1}{\alpha}}}{\sum_{b=1}^K(\hat{Z}_{b,t+1} + \sigma_{b,t+1})^{\frac{1}{\alpha}}}, \tag{5.37}$$

define $Z_a := Z\pi_a^{(\alpha)}$, this leads to the regret

$$R(q_t) = \frac{1}{\alpha(\alpha-1)}\left(\sum_a \hat{q}_{at}^{1-\alpha}\pi_a^{(\alpha)} - (\sum_a(\pi_a^{(\alpha)})^{\frac{1}{\alpha}})^\alpha\right) \tag{5.38}$$

$$= \frac{1}{\alpha(\alpha-1)Z}\left(\sum_a \hat{q}_{at}^{1-\alpha}Z_a - (\sum_a Z_a^{\frac{1}{\alpha}})^\alpha\right). \tag{5.39}$$

Theorem 8. *The (instantaneous) regret for general α-loss for $\alpha \in (0,2]$ is upper bounded by* $\frac{2^{\frac{1}{2\alpha}+2}c^{\frac{1}{2\alpha}+1}K^{\frac{1}{2}}}{\alpha^2 Z\sqrt{t}}\left(\sum_a \tau_a Z_a^{\frac{1}{2\alpha}-1}\right)\sqrt{\sum_a \tau_a^{\frac{1}{\alpha}}}(\log t)^{\frac{1}{2}(\frac{1}{2\alpha}+1)}$.

Recall that with proposal

$$q_{a,t+1} = \frac{(\hat{Z}_{a,t+1} + \sigma_{a,t+1})^{\frac{1}{\alpha}}}{\sum_{b=1}^{K}(\hat{Z}_{b,t+1} + \sigma_{b,t+1})^{\frac{1}{\alpha}}} \tag{5.40}$$

where $\sigma_{at} = c\tau_a\sqrt{\frac{\log t}{N_{at}}}$, $\hat{Z}_{a,t+1} := \frac{\sum_{l\leq t:A_l=a} Y_{al}}{N_{a,t+1}}$, N_{at} is the number of times arm a has been picked up until time t, and $Y_{al} := \left(\frac{f(x_l)}{g_a(x_l)}\right)^{\alpha}$, we would like to bound the expectation of regret $R(q_t) = \frac{1}{\alpha(\alpha-1)Z}\left(\sum_a \hat{q}_{at}^{1-\alpha} Z_a - (\sum_a Z_a^{\frac{1}{\alpha}})^{\alpha}\right)$, where $Z_a = \int_{\mathcal{X}_a}\left(\frac{f(x)}{g_a(x)}\right)^{\alpha} g_a(x)dx$.

Proof. We follow the same proof structure as in Theorem 5. Recall

$$B_t := \{|\hat{Z}_{as} - Z_{as}| < \sigma_{as} \ \ \forall 1 \leq a \leq K, \left[\frac{t}{2}\right] < s \leq t\} \tag{5.41}$$

$$C_t := \{N_{at} > \beta_{at} \ \forall\ 1 \leq a \leq K\}, \tag{5.42}$$

and we re-define $\beta_{at} := \frac{tZ_a^{\frac{1}{\alpha}}}{4(\sum_a (Z_a + 2c\tau_a\sqrt{\log t})^{\frac{1}{\alpha}}}$. The expected reward at time t therefore can be written as:

$$\mathbb{E}[R_t] = \mathbb{E}[R_t|B_t \cap C_t]\mathbb{P}(B_t \cap C_t) + \mathbb{E}[R_t|B_t \cap C_t^c]\mathbb{P}(B_t \cap C_t^c) + \mathbb{E}[R_t|B_t^c]\mathbb{P}(B_t^c). \tag{5.43}$$

For the first term in (5.43), we have

$$\mathbb{E}[R_t|B_t \cap C_t]\mathbb{P}(C_t|B_t)\mathbb{P}(B_t) \leq \mathbb{E}[R_t|N_{at} \geq \beta_{at}\forall a, B_t] \tag{5.44}$$

$$\leq \frac{1}{Z\alpha(\alpha-1)}\mathbb{E}\left[\frac{\sum_a Z_a^{\frac{1}{\alpha}}}{\left(\sum_a (Z_a + 2\sigma_{at})^{\frac{1}{\alpha}}\right)^{1-\alpha}} - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right] \tag{5.45}$$

$$= \frac{1}{Z\alpha(\alpha-1)}\mathbb{E}\left[\left(\sum_a Z_a^{\frac{1}{\alpha}}\left(\sum_a (Z_a + 2\sigma_{at})^{\frac{1}{\alpha}}\right)^{\alpha-1} - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right)\right]. \tag{5.46}$$

$$\tag{5.47}$$

We apply Taylor's expansion twice to functions $h(x) = x^{\frac{1}{\alpha}}$ and $h(x) = x^{\alpha-1}$ to get

$$\left(\sum_a (Z_a + 2\sigma_{at})^{\frac{1}{\alpha}}\right)^{\alpha-1} = \left(\sum_a (Z_a^{\frac{1}{\alpha}}) + \frac{2}{\alpha}\sum_a Z_a^{\frac{1}{\alpha}-1}\sigma_{at} + \mathcal{O}(\sigma_{at}^2)\right)^{\alpha-1} \tag{5.48}$$

$$= \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1} + \frac{2(\alpha-1)}{\alpha}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-2}\left(\sum_a Z_a^{\frac{1}{\alpha}-1}\sigma_{at}\right) + \mathcal{O}(\sigma_{at}^2), \tag{5.49}$$

hence

$$(5.46) = \frac{1}{Z\alpha(\alpha-1)}\mathbb{E}\left[\frac{2(\alpha-1)}{\alpha}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a Z_a^{\frac{1}{\alpha}-1}\sigma_{at}\right) + \mathcal{O}(\sigma_{at}^2)\right] \tag{5.50}$$

$$\leq \frac{2c}{Z\alpha}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a \tau_a Z_a^{\frac{1}{\alpha}-1}\sqrt{\frac{\log t}{\beta_{at}}}\right) + \mathcal{O}\left(\tau_a^2\frac{\log t}{\beta_{at}}\right) \tag{5.51}$$

$$= \frac{2c}{Z\alpha^2}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\sum_a \tau_a Z_a^{\frac{1}{\alpha}-1}\sqrt{\frac{4\log t\sum_a(Z_a+2c\tau_a\sqrt{\log t})^{\frac{1}{\alpha}}}{tZ_a^{\frac{1}{\alpha}}}} + \textit{lower order term} \tag{5.52}$$

$$\leq \frac{4c}{Z\alpha^2\sqrt{t}}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\sum_a \tau_a Z_a^{\frac{1}{2\alpha}-1}\sqrt{\log t\left(\sum_a Z_a^{\frac{1}{\alpha}} + \sum_a (2c\tau_a)^{\frac{1}{\alpha}}(\log t)^{\frac{1}{2\alpha}}\right)} \tag{5.53}$$

$$+ \textit{lower order term} \tag{5.54}$$

$$= \frac{2^{\frac{1}{2\alpha}+2}c^{\frac{1}{2\alpha}+1}K^{\frac{1}{2}}}{\alpha^2 Z\sqrt{t}}\left(\sum_a \tau_a Z_a^{\frac{1}{2\alpha}-1}\right)\sqrt{\sum_a \tau_a^{\frac{1}{\alpha}}(\log t)^{\frac{1}{2}(\frac{1}{2\alpha}+1)}} + \textit{lower order term.} \tag{5.55}$$

We proceed by bounding the probability $\mathbb{P}(C_t^c|B_t)$. Conditioning on B_t and for any $\frac{t}{2} < s < t$, we have

$$q_{as} = \frac{\left(\hat{Z}_{as}+\sigma_{as}\right)^{\frac{1}{\alpha}}}{\sum_b\left(\hat{Z}_{bs}+\sigma_{bs}\right)^{\frac{1}{\alpha}}} \geq \frac{Z_a^{\frac{1}{\alpha}}}{\sum_b(Z_b+2\sigma_{bs})^{\frac{1}{\alpha}}} \quad \textit{by definition of } B_t \tag{5.56}$$

$$\implies \sum_{s=1}^{t} q_{as} \geq \sum_{s=\frac{t}{2}+1}^{t} q_{as} \geq \frac{tZ_a^{\frac{1}{\alpha}}}{2\sum_a(Z_a+2c\tau_a\sqrt{\log t})^{\frac{1}{\alpha}}} = 2\beta_{at} \quad \textit{since } s \leq t \textit{ and } N_{as} \geq 1 \tag{5.57}$$

$$\implies \mathbb{P}\left(N_{at} < \beta_{at}|B_t\right) \leq \mathbb{P}\left(N_{at} < \frac{1}{2}\sum_{s=1}^{t} q_{as}|B_t\right) \leq 2\exp\left(-\frac{1}{2t}\beta_{at}^2\right) \quad \textit{by lemma 7.} \tag{5.58}$$

Hence for the second term of (5.43), we have

$$\mathbb{E}[R_t|B_t\cap C_t^c]\mathbb{P}(C_t^c|B_t)\mathbb{P}(B_t) \tag{5.59}$$

$$\leq \frac{2}{Z\alpha(\alpha-1)}\mathbb{E}\left[\left(\frac{2(\alpha-1)}{\alpha}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a Z_a^{\frac{1}{\alpha}-1}\sigma_{at}\right)+\mathcal{O}(\sigma_{at}^2)\right)|B_t\cap C_t^c\right] \tag{5.60}$$

$$\cdot\sum_a \exp\left(-\frac{1}{2t}\beta_{at}^2\right) \tag{5.61}$$

$$\leq \frac{4}{Z\alpha^2}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a Z_a^{\frac{1}{\alpha}-1}\mathbb{E}[\sigma_{at}|B_t\cap C_t^c]\right)\sum_a\frac{2t}{\beta_{at}^2}+\textit{lower order term} \tag{5.62}$$

$$\leq \frac{4c\sqrt{\log t}\sum_a\tau_a}{Z\alpha^2}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a Z_a^{\frac{1}{\alpha}-1}\right)\sum_a\frac{32\left(\sum_a(Z_a+2c\tau_a\sqrt{\log t})^{\frac{1}{\alpha}}\right)^2}{tZ_a^{\frac{2}{\alpha}}} \tag{5.63}$$

$$+\textit{lower order term} \tag{5.64}$$

$$\leq \frac{2^{7+\frac{2}{\alpha}}c^{1+\frac{2}{\alpha}}(\log t)^{\frac{1}{2}+\frac{1}{\alpha}}\sum_a\tau_a(\sum_a\tau_a^{\frac{1}{\alpha}})^2}{\alpha^2tZ}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a Z_a^{\frac{1}{\alpha}-1}\right)\sum_a Z_a^{-\frac{2}{\alpha}} \tag{5.65}$$

$$+\textit{lower order term.} \tag{5.66}$$

The proof of $\mathbb{P}(B_t^c)$ being small is the same as that in theorem 5. Recall

$$\mathbb{P}(B_t^c)\leq\sum_{a=1}^K\sum_{s=\frac{t}{2}}^t\mathbb{P}(|\hat{Z}_{as}-Z_a|>\sigma_{as})\leq\frac{2K}{t}. \tag{5.67}$$

Therefore for the third term in (5.43), we have

$$\mathbb{E}[R_t|B_t^c]\mathbb{P}(B_t^c)$$

$$\leq\frac{2K}{\alpha(\alpha-1)Zt}\mathbb{E}\left[\left(\sum_a(\hat{Z}_{at}+\sigma_{at})^{\frac{1-\alpha}{\alpha}}Z_a\left(\sum_a(\hat{Z}_{at}+\sigma_{at})^{\frac{1}{\alpha}}\right)^{\alpha-1}\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right)|B_t^c\right].$$

$$
\begin{aligned}
&\leq \frac{2K}{\alpha(\alpha-1)Zt}\left(\sum_a (c\tau_a\sqrt{\log t})^{\frac{1-\alpha}{\alpha}} Z_a \mathbb{E}\left[\left(\sum_a (\hat{Z}_{at}+\sigma_{at})^{\frac{1}{\alpha}}\right)^{\alpha-1}|B_t^c\right] - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right)\\
&\leq \frac{2K}{\alpha(\alpha-1)Zt}\left(\sum_a (c\tau_a\sqrt{\log t})^{\frac{1-\alpha}{\alpha}} Z_a \left(\sum_a \mathbb{E}[(\hat{Z}_{at}+\sigma_{at})^{\frac{1}{\alpha}}|B_t^c]\right)^{\alpha-1} - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right) \quad \textit{by Jensen}\\
&\leq \frac{2K}{\alpha(\alpha-1)Zt}\left(\sum_a (c\tau_a\sqrt{\log t})^{\frac{1-\alpha}{\alpha}} Z_a \left(\sum_a (\mathcal{O}(t\sqrt{\log t})+Z_a)^{\frac{1}{\alpha}}\right)^{\alpha-1} - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right)\\
&\leq \frac{2K}{\alpha(\alpha-1)Zt}\left(\sum_a (c\tau_a\sqrt{\log t})^{\frac{1-\alpha}{\alpha}} Z_a \left(\sum_a (Z_a^{\frac{1}{\alpha}}+\mathcal{O}((t\sqrt{\log t})^{\frac{1}{\alpha}})\right)^{\alpha-1} - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right)\\
&\leq \frac{2K}{\alpha(\alpha-1)Zt}\left(\sum_a (c\tau_a\sqrt{\log t})^{\frac{1-\alpha}{\alpha}} Z_a \left(\left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha-1}+\mathcal{O}((t\sqrt{\log t})^{\frac{\alpha-1}{\alpha}})\right) - \left(\sum_a Z_a^{\frac{1}{\alpha}}\right)^{\alpha}\right)\\
&= \frac{2Kc^{\frac{1-\alpha}{\alpha}}\sum_a \tau_a^{\frac{1-\alpha}{\alpha}}(\sum_a Z_a)}{\alpha(\alpha-1)Z} t^{-\frac{1}{\alpha}} + \textit{lower order term.}
\end{aligned}
$$

Summing over t gives the cumulative regret is dominated by the first term in (5.43) which is $\mathcal{O}(t^{-\frac{1}{2}}(\log t)^{\frac{1}{2}(\frac{1}{2\alpha}+1)})$.

□

Chapter 6

Discussion

6.1 Conclusion

Uncertainty quantification remains an important aspect of machine learning, which can not be fully addressed by deep learning despite its popularity and state-of-the-art performance on various benchmark tasks. We have focused on probabilistic machine learning and Bayesian methods which enable calibrated uncertainties to be computed.

We have discussed Bayesian modelling in the first half of the thesis and inference in the latter half. In particular, we began with the discussion of Gaussian Process, which is a class of flexible Bayesian non-parametric model. However, its cubic scaling with respect to the number of observations makes it infeasible for large dataset. From the weight-space of GP, the function value $f(x)$ can be viewed as the dot product between θ and a feature map $\phi(x)$, where θ has a Gaussian prior and $\phi(x)$ satisfies $k(x, x') =< \phi(x), \phi(x') >$ and k is the kernel associated with f. If the kernel is a product of kernels across different dimensions $d = 1, \cdots, D$, the feature map can be written as the kronecker product of the feature maps in each dimension. This motivated us to propose the Tucker Gaussian Process, which finds a low-rank approximation of θ using Tucker decomposition. The feature maps are chosen according to different applications: 1) for grid structured data, one can compute the features from cholesky decomposition; 2) for general data with stationary kernel, one can use Fourier feature maps; 3) in collaborative filtering, one can use the identity feature. When the side information about users or items are available, TGP has a natural way of incorporating the side information into the feature map. We have applied the model to regression problem on spatial data, spatial-temporal data and on the MovieLens recommendation dataset, which have shown promising results.

In chapter 3, we have proposed a Bayesian optimisation method on combinatorial space. Bayesian optimisation is a black-box optimiser but can not be naively applied

to combinatorial space. The idea is to learn a low dimensional representation of the original input space using a variational auto-encoder. The application we considered is automatic statistician, in which the goal is to select the kernel in Gaussian Process models given any dataset. We randomly generate a number of kernel representations based on compositions (replacement, addition and multiplication) of four base kernels, we then use a context free grammar to represent each kernel k with a vector of binary digits. In order to take into consideration of the dataset for which we need to search for the best kernel, we append to the kernel representation a vector describing the difference between the Gram matrices of the kernel k and of the base kernels. A VAE is then used to learn a low dimensional representation, after which Bayesian optimisation is performed on the continuous low dimensional space to find the best kernel combination based on some evaluation criterion (we have used test RMSE). We have shown experimentally that our method outperforms the greedy search algorithm on a range of datasets. We have also applied our algorithm on a natural scene understanding example, where the goal is to search for the XML description given an input image. We have simplified a complicated computer vision problem as an optimisation problem, and our algorithm is able to produce sensible images.

For the inference part, we have developed Relativistic Hamiltonian Monte Carlo (RHMC) algorithm in chapter 4, which is a variant of HMC that is more robust to changes of parameters. The idea is to replace Newton's kinetic energy with Einstein's relativistic kinetic energy. By doing so, the speed of the parameters is upper limited by the speed of light, which is a tuning parameter. By taking the limit as the temperature goes to zero, we reach an optimisation algorithm that is closely related to various optimisation algorithms such as Adam, Adagrad, RMSprop, etc. All of these use pre-conditioning of the gradients, which adaptively change the stepsize in different dimensions. Our optimisation algorithm differs in that we have an auxiliary momentum p that serves as reservoirs of previous gradient computations, thus can integrate and smooth out gradient signals from previous mini-batches of data. We further extend our method to a thermostat version, which stabilise momentum fluctuations and remedy the problem of the unknown noise coming from the stochastic gradients. We have applied RHMC and its stochastic variants to a range of experiments, including the banana shaped example, Gaussian mixture models and logistic regression. We show that they outperformed Newtonian counterparts. Our optimisation algorithm is competitive compared with Adam on the MNIST dataset.

In chapter 5, we have introduced a new approach in adaptive importance sampling algorithm called Daisee. It is among the frontier work which considers the exploration

and exploitation tradeoff in adaptive importance sampling. In particular, we borrow the idea from multi-armed bandit problem and construct a proposal distribution using upper confidence bound. We first partition the sample space into subspaces, where each subspace is viewed as an arm, and the rewards for each arm are related to the importance weights obtained from that subspace. We prove theoretically that our proposed algorithm Daisee has a sub-linear growth of cumulative regret, where the regret is based on the KL divergence between the target and the proposal distributions. We also extended our method to the hierarchical case (HiDaisee), where instead of fixing the partition, a better partitioning is constructed by finely graining the area with highly fluctuated density. We have demonstrated sets of experiments to verify our theoretical results for the non hierarchical case, and showed promising results on the hierarchical case with a challenging banana shaped example.

6.2 Future Work

Our research in this thesis opens the doors to many interesting research directions. We will outline a few of these below.

6.2.1 Adaptive MCMC

The adaptive importance sampling scheme opens up a new array of research that can be extended and studied in other MCMC algorithms. In particular, we investigated the importance sampling from the exploration-exploitation trade off perspective and viewed it as an online learning problem that is closely related to reinforcement learning. The theoretical results showed that the pseudo regret grows sub-linearly with time, which is further tested in the experiment section. It is interesting to study similar results for other elaborate MCMC schemes which can improve the convergence of the Markov chains in inference and can be applied widely in a range of applications, from deep learning to traditional statistical learning problems.

6.2.2 Transfer Learning

The SVO model we have discussed in chapter 3 can be further extended in the context of transfer learning [Torrey and Shavlik, 2010, Pan et al., 2010]. Suppose one is interested in choosing the optimal kernel for a dataset, and he has some knowledge about similar datasets which have already been studies before. Then he can leverage the information about which kernels worked well on those previous datasets to build

the prior distribution to generate all the kernel combinations, before learning the low dimensional space using VAE. By leveraging previous knowledge, one can learn a latent space in the space of kernels which are more likely to work well for this dataset, and makes the optimisation procedure thereafter more efficient.

6.3 Closing Remarks

We believe that the development of artificial intelligence contributes to the research community and leads to better life for the human being. In particular, the research into probabilistic machine learning with calibrated uncertainties plays an important role in making AI reliable and interpretable. There are many interesting unsolved problems in probabilistic machine learning such as developing efficient and flexible models, and improving accuracy of approximate inference. It is our mission to continue working on flexible modelling, accurate approximate inference and efficient optimization to improve the current framework.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. 2016.

Gabriele Abbati, Alessandra Tosi, Michael A Osborne, and Seth Flaxman. Adageo: Adaptive geometric learning for optimization and sampling. In *NIPS workshop in Approximate inference*, 2017.

D. Agarwal and B. Chen. Regression-based latent factor models. In *ACM SIGKDD*, 2009.

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, pages 39–1, 2012.

Janis Aldins, Stanley J Brodsky, Andrew J Dufner, and Toichiro Kinoshita. Photon-photon scattering contribution to the sixth-order magnetic moments of the muon and electron. *Physical Review D*, 1(8):2378, 1970.

Shun-Ichi Amari. Differential geometry of curved exponential families-curvatures and information loss. *The Annals of Statistics*, pages 357–385, 1982.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Per-Arne Andersen, Morten Goodwin, and Ole-Christoffer Granmo. The dreaming variational autoencoder for reinforcement learning environments. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 143–155. Springer, 2018.

C. Andrieu and A. Doucet. Simulated annealing for maximum a posteriori parameter estimation of hidden markov models. *Information Theory, IEEE Transactions on*, 46(3):994–1004, 2000.

Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. *arXiv preprint arXiv:1702.07825*, 2017.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

Akshay Balsubramani. Sharp finite-time iterated-logarithm martingale concentration. *arXiv preprint arXiv:1405.2639*, 2014.

Matthew James Beal et al. *Variational algorithms for approximate Bayesian inference*. university of London London, 2003.

James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

Håvard Berland. Automatic differentiation. 2006.

Donald A Berry and Bert Fristedt. *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. Springer, 1985.

A. Beskos, N. Pillai, G. O. Roberts, J. M. Sanz-Serna, and A. M. Stuart. Optimal tuning of hybrid Monte Carlo algorithm. *Bernoulli*, 19:1501–1534, 2013. doi: 10.3150/12-BEJ414. URL http://dx.doi.org/10.3150/12-BEJ414.

G. Beylkin, J. Garcke, and M. Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing*, 2009.

John B Biggs. The role of metalearning in study processes. *British journal of educational psychology*, 55(3):185–212, 1985.

D. Billsus and M. Pazzani. Learning collaborative information filters. In *ICML*, 1998.

Christopher M Bishop. Pattern recognition and machine learning (information science and statistics) springer-verlag new york. *Inc. Secaucus, NJ, USA*, 2006.

Thomas E Booth and John S Hendricks. Importance estimation in forward monte carlo calculations. *Nuclear Technology-Fusion*, 5(1):90–100, 1984.

George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990. ISBN 0816211043.

R. Bro. PARAFAC. Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 1997.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Sébastien Bubeck, Nicolo Cesa-Bianchi, and Gábor Lugosi. Bandits with heavy tail. *IEEE Transactions on Information Theory*, 59(11):7711–7717, 2013.

Monica F Bugallo, Victor Elvira, Luca Martino, David Luengo, Joaquin Miguez, and Petar M Djuric. Adaptive importance sampling: the past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79, 2017.

M.P. Calvo. *Numerical hamiltonian problems*, volume 7. CRC Press, 1994.

Olivier Cappé, Arnaud Guillin, Jean-Michel Marin, and Christian P Robert. Population monte carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004.

Olivier Cappé, Randal Douc, Arnaud Guillin, Jean-Michel Marin, and Christian P Robert. Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 18(4):447–459, 2008.

B. Carpenter. Stan: A probabilistic programming language. *Journal of Statistical Software*, 2015.

Alexandra Carpentier and Rémi Munos. Finite time analysis of stratified sampling for monte carlo. In *Advances in Neural Information Processing Systems*, pages 1278–1286, 2011.

Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *arXiv preprint arXiv:1511.01437*, 2015.

T. Chen, E. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1683–1691, 2014a. URL http://jmlr.org/proceedings/papers/v32/cheni14.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Yutian Chen, Vikash Mansinghka, and Zoubin Ghahramani. Sublinear-time approximate mcmc transitions for probabilistic programs. *arXiv preprint arXiv:1411.1690*, 2014b.

H. Chipman, E. George, and R. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 2010.

Andrzej Cichocki and Shun-ichi Amari. Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568, 2010.

John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *arXiv preprint arXiv:1806.02813*, 2018.

Jean Cornuet, JEAN-MICHEL MARIN, Antonietta Mira, and Christian P Robert. Adaptive multiple importance sampling. *Scandinavian Journal of Statistics*, 39(4): 798–812, 2012.

DR Cox and DV Hinkley. Theoretical statistics chapman and hall, london. *See Also*, 1974.

Zhenwen Dai, Andreas Damianou, Javier González, and Neil Lawrence. Variational auto-encoded deep Gaussian processes. *International Conference on Learning Representations (ICLR)*, 2016.

Andreas C. Damianou, Michalis K. Titsias, and Neil D. Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research*, 17(42):1–62, 2016.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

Li Deng. An overview of deep-structured learning for information processing. 2011.

Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

Mark Denny. Introduction to importance sampling in rare-event simulations. *European Journal of Physics*, 22(4):403, 2001.

N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. Bayesian sampling using stochastic gradient thermostats. In *NIPS*, pages 3203–3211, 2014. URL http://papers.nips.cc/paper/5592-bayesian-sampling-using-stochastic-gradient-thermostats.

P. Drineas and M. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JMLR*, 2005.

S. Duane, A.D. Kennedy, B.J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *arXiv preprint arXiv:1302.4922*, 2013.

Albert Einstein. Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, 322(10):891–921, 1905. doi: http://dx.doi.org/10.1002/andp.19053221004.

Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *arXiv preprint arXiv:1805.08498*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

Seth Flaxman, Andrew Wilson, Daniel Neill, Hannes Nickisch, and Alex Smola. Fast kronecker inference in gaussian processes with non-gaussian likelihoods. In *International Conference on Machine Learning*, pages 607–616, 2015.

Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and SM Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

A. Gelman and D. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 1992.

S. Geman and C. Hwang. Diffusions for global optimization. *SIAM Journal on Control and Optimization*, 24(5):1031–1043, 1986.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

Zoubin Ghahramani and Matthew J Beal. Variational inference for bayesian mixtures of factor analysers. In *Advances in neural information processing systems*, pages 449–455, 2000.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, March 2011. doi: 10.1111/j.1467-9868.2010.00765.x. URL http://dx.doi.org/10.1111/j.1467-9868.2010.00765.x.

Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13, 2016.

Javier Gonzalez, Joseph Longworth, David C. J James, and Neil D. Lawrence. Bayesian optimization for synthetic gene design. In *NIPS workshop in Bayesian optimization*, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

J. Gorham and L. W. Mackey. Measuring Sample Quality with Stein's Method. In *NIPS*, pages 226–234, 2015. URL http://papers.nips.cc/paper/5768-measuring-sample-quality-with-steins-method.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

Karol Gregor and Frederic Besse. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582, 2016.

Roger Grosse, Ruslan R Salakhutdinov, William T Freeman, and Joshua B Tenenbaum. Exploiting compositionality to explore a large space of model structures. *arXiv preprint arXiv:1210.4856*, 2012.

H. Haario, E. Saksman, and J. Tamminen. Haario. *Computational Statistics*, 14(3): 375–396, 1999. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.3205&rep=rep1&type=pdf.

P. Hall and C. Heyde. *Martingale Limit Theory and its Application*. Academic press, 2014.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Hera Y He and Art B Owen. Optimal mixture weights in multiple importance sampling. *arXiv preprint arXiv:1411.3954*, 2014.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pages 918–926, 2014.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

M. Hoffman and A. Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, 2014.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. ISBN 0321455363.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jonathan H Huggins, Trevor Campbell, Mikołaj Kasprzak, and Tamara Broderick. Scalable gaussian process inference with finite-data mean and variance guarantees. *arXiv preprint arXiv:1806.10234*, 2018.

Yunseong Hwang and Jaesik Choi. The automatic statistician: A relational perspective. *CoRR*, abs/1511.08343, 2015.

M. Imaizumi and K. Hayashi. Doubly decomposing nonparametric tensor regression. In *ICML*, 2016.

Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1655–1664, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Tang Jie and Pieter Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, pages 1000–1008, 2010.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Structured vaes: Composing probabilistic graphical models and variational autoencoders. *arXiv preprint arXiv:1603.06277*, 2, 2016.

Johnson-Lindenstrauss. Johnson-lindenstrauss theory. URL http://lear.inrialpes.fr/people/harchaoui/teaching/2013-2014/ensl/m2/lecture6.pdf.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

Herman Kahn and Theodore E Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

Mohammad Emtiyaz Khan and Didrik Nielsen. Fast yet simple natural-gradient descent for variational inference in complex models. *arXiv preprint arXiv:1807.04489*, 2018.

Hyunjik Kim and Yee Whye Teh. Scalable structure discovery for regression using gaussian processes. *AutoML 2016 Proceedings, Journal of Machine Learning Research Workshop and Conference Proceedings*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014a. URL http://arxiv.org/abs/1412.6980.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014b.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Thomas Kollig and Alexander Keller. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 245–257. Springer, 2006.

Wouter M. Koolen. A quick and dirty finite time law of the iterated logarithm result. *http://blog.wouterkoolen.info/QnD_LIL/post.html*, 2017.

Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81:1–10, 2009.

Frank R. Kschischang. The complementary error function. *http://www.comm.utoronto.ca/frank/notes/erfc.pdf*, 2017.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954. JMLR. org, 2017.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

N. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *NIPS*, 2004.

N. Lawrence and R. Urtasun. Non-linear matrix factorization with Gaussian processes. In *ICML*, 2009.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436, 2015.

Hyungtae Lee, Sungmin Eum, and Heesung Kwon. Me r-cnn: multi-expert region-based cnn for object detection. *CoRR*, 2017.

B. Leimkuhler and S. Reich. A metropolis adjusted nosé-hoover thermostat. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43:743–755, 7 2009. ISSN 1290-3841. doi: 10.1051/m2an/2009023. URL http://www.esaim-m2an.org/article_S0764583X09000235.

G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192–203, 1978.

Florian Leprêtre, Fabien Teytaud, and Julien Dehos. Multi-armed bandit for stratified sampling: Application to numerical integration. In *TAAI 2017-Conference on Technologies and Applications of Artificial Intelligence*, 2017.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

C. Li, C. Chen, D. Carlson, and L. Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. *arXiv preprint arXiv:1512.07666*, 2015.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pages 3812–3822, 2017.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

X. Lu, T. Rainforth, Y. Zhou, J.-W. van de Meent, and Y. Whye Teh. On Exploration, Exploitation and Learning in Adaptive Importance Sampling. *ArXiv e-prints*, October 2018.

Y. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient MCMC. In *NIPS*, 2015.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Gustavo Malkomes, Charles Schaff, and Roman Garnett. Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems*, pages 2900–2908, 2016.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Luca Martino, Victor Elvira, David Luengo, and Jukka Corander. An adaptive population importance sampler. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 8038–8042. IEEE, 2014.

Luca Martino, Victor Elvira, David Luengo, and Jukka Corander. Layered adaptive importance sampling. *Statistics and Computing*, 27(3):599–623, 2017.

Donald B Maudsley. A theory of meta-learning and principles of facilitation: An organismic perspective. 1980.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Nicolas Meuleau, Leonid Peshkin, Leslie P Kaelbling, and Kee-Eung Kim. Off-policy policy search. *MIT Articical Intelligence Laboratory*, 2000.

Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pages 6246–6256, 2018.

Jonas Mockus. On bayesian methods for seeking the extremum and their application. In *IFIP Congress*, pages 195–200, 1977.

R. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2011.

R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

Radford M Neal. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Citeseer, 1992.

Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.

James Neufeld, András György, Dale Schuurmans, and Csaba Szepesvári. Adaptive monte carlo via bandit allocation. *arXiv preprint arXiv:1405.3318*, 2014.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

Art B. Owen. *Monte Carlo theory, methods and examples.* 2013.

Art B Owen, Yury Maximov, and Michael Chertkov. Importance sampling the union of rare events with an application to power systems analysis. *arXiv preprint arXiv:1710.06965*, 2017.

John Paisley, David Blei, and Michael Jordan. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

S. Park, Y. Kim, and S. Choi. Hierarchical bayesian matrix factorization with side information. In *IJCAI*, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

S. Patterson and Y. W. Teh. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *NIPS*, pages 3102–3110, 2013. URL http://dblp.uni-trier.de/db/conf/nips/nips2013.html#PattersonT13.

Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.

Martin Piotte and Martin Chabbert. The pragmatic theory solution to the netflix grand prize. *Netflix prize documentation*, 2009.

I. Porteous and M. Welling. Bayesian matrix factorization with side information and Dirichlet process mixtures. In *AAAI*, 2010.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Tom Rainforth, Yuan Zhou, Xiaoyu Lu, Yee Whye Teh, Frank Wood, Hongseok Yang, and Jan-Willem van de Meent. Inference trees: Adaptive inference with exploration. *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

C. Rasmussen and H. Nickisch. Gaussian Processes for Machine Learning (GPML) toolbox, 2010.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

Stochastic Relaxation. Gibbs distributions, and the bayesian restoration of images. s. geman and d. geman. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

Gerardo Rubino and Bruno Tuffin. *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

W. Rudin. Fourier analysis on groups. *AMS*, 1964.

Y. Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, 2011.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2008a.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov Chain Monte Carlo. In *ICML*, 2008b.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Rajat Sen, Karthikeyan Shanmugam, and Sanjay Shakkottai. Contextual bandits with stochastic experts. *arXiv preprint arXiv:1802.08737*, 2018.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

X. Shang, Z. Zhu, B. Leimkuhler, and A. J. Storkey. Covariance-Controlled Adaptive Langevin Thermostat for Large-Scale Bayesian Sampling. In *NIPS*, pages 37–45, 2015. URL https://papers.nips.cc/paper/5978-covariance-controlled-adaptive-langevin-thermostat-for-large-scale-bayesia

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

A. Singh and G. Gordon. Relational learning via collective matrix factorization. In *ACM SIGKDD*, 2008.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, 2005.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Stan Development Team. Stan: A C++ library for probability and sampling, version 1.0, 2012.

Joshua Susskind, Volodymyr Mnih, Geoffrey Hinton, et al. On deep generative models with applications to recognition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2857–2864. IEEE, 2011.

T. Suzuki, H. Kanagawa, H. Kobayashi, N. Shimizu, and Y. Tagami. Minimax optimal alternating minimization for kernel nonparametric tensor learning. In *NIPS*, 2016.

Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, and Michael A. Osborne. Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. In *NIPS workshop on Bayesian Optimization in theory and practice (BayesOpt?13)*, 2013.

Y. W. Teh, L. Hasenclever, T. Lienart, S. Vollmer, S. Webb, B. Lakshminarayanan, and C. Blundell. Distributed bayesian learning with stochastic natural-gradient expectation propagation and the posterior server. *CoRR*, abs/1512.09327, 2015. URL http://arxiv.org/abs/1512.09327.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, 2009.

Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010.

Andreas Töscher, Michael Jahrer, and Robert M Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, pages 1–52, 2009.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent

variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636, 2017.

George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for monte carlo objectives. *arXiv preprint arXiv:1810.04152*, 2018.

L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 1966.

Herke van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3928–3934. IEEE, 2016.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016a.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *J. Artif. Int. Res.*, 55(1):361–387, January 2016b. ISSN 1076-9757.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, 2009.

M. Welling and Y.W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *ICML*, pages 681–688. Omnipress, 2011a.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011b.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, 2001.

Andrew G Wilson, Elad Gilboa, Arye Nehorai, and John P Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*, pages 3626–3634, 2014.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

Jiajun Wu, Joshua B. Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *CVPR*, 2017.

Tailin Wu, John Peurifoy, Isaac L Chuang, and Max Tegmark. Meta-learning autoencoders for few-shot prediction. *arXiv preprint arXiv:1807.09912*, 2018.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

T. Xifara, C. Sherlock, S. Livingstone, S. Byrne, and M. Girolami. Langevin diffusions and the metropolis-adjusted langevin algorithm. *Statistics & Probability Letters*, 91 (C):14–19, 2014. URL http://EconPapers.repec.org/RePEc:eee:stapro:v:91:y:2014:i:c:p:14-19.

Z. Xu, F. Yan, and Y. Qi. Infinite Tucker decomposition: Nonparametric bayesian models for multiway data analysis. *arXiv:1108.6296*, 2011.

Emre Yamangil, Russell Bent, and Scott Backhaus. Designing resilient electrical distribution grids. *arXiv preprint arXiv:1409.4477*, 2014.

K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu. Stochastic relational models for discriminative link prediction. In *NIPS*, 2006.

Scott L Zeger and M Rezaul Karim. Generalized linear models with random effects; a gibbs sampling approach. *Journal of the American statistical association*, 86(413): 79–86, 1991.

Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

Q. Zhao, L. Zhang, and A. Cichocki. A tensor-variate gaussian process for classification of multidimensional structured data. In *AAAI*, 2013.

S. Zhe, Y. Qi, Y. Park, Z. Xu, I. Molloy, and S. Chari. Dintucker: Scaling up gaussian process models on large multidimensional arrays. In *AAAI*, 2016a.

S. Zhe, K. Zhang, P. Wang, K. Lee, Z. Xu, Y. Qi, and Z. Ghahramani. Distributed flexible nonlinear tensor factorization. In *NIPS*, 2016b.