Structured Bayesian Approximate Inference

Bonnevie, Rasmus

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Structured Bayesian Approximate Inference

Rasmus Bonnevie

Kongens Lyngby 2018

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

This thesis seeks to investigate different facets of the class of Bayesian probabilistic models where the random variables exhibit strong dependencies and simultaneously lack any conditional independence structure, preventing the distribution from being factorized.

Without a tractable factorization, a lot of standard inference algorithms become unavailable. We consider the application of variational inference from two different perspectives. In the first scenario we start with an extended model with conditional independence structure, and try to take the auxiliary parameters out of the equation in an optimal manner in a process emulating marginalization. In the second scenario, we tackle the variational problem directly, trying to find a suitable way to represent unfactorized models in an efficient manner, by introducing a separate form of structure. For discrete models, we find good approximations in the tensor literature that can model structure without sacrificing tractability.

Finally, we consider a problem involving Gaussian processes that take random variables as input, leading to an inefficient inference problem. We develop a procedure that allows the stochastic component of the random input to be integrated into the kernel of the Gaussian process.

Summary (Danish)

Denne afhandling ønsker at undersøge forskellige facetter af en klasse af Bayesianske probabilistiske modeller, hvor at de stokastiske variable har indbyrdes høj afhængighed, mens at modellen samtidig mangler konditioneret uafhængighedsstruktur, hvilket forhindrer fordelingen i at blive faktoriseret.

Uden en beregningsvenlig faktorisering kommer de fleste almindelige inferensalgoritmer til kort. Vi overvejer, hvordan variationel inferens kan bruges ud fra to forskellige perspektiver. I det første scenarie benytter vi en udvidet model med konditioneret uafhængighedsstruktur og prøver at fjerne de overflødige variable på en optimal måde, sådan at processen emulerer marginalisering. I det andet scenarie håndterer vi det variationelle problem direkte og prøver at finde en effektiv måde at repræsentere ufaktoriserede modeller ved at introducere en helt tredje form for struktur for at øge effektiviteten. I tilfældet med diskrete modeller finder vi meget effektive approksimationer i tensor-litteraturen, som kan modellere struktur uden at gå på kompromis med den beregningsmæssige effektivitet.

Endelig undersøger vi et problem med gaussiske processer som tager stokastiske variable som input, hvilket leder til et tungt inferensproblem. Vi udvikler en procedure der tillader den stokastiske komponent i inputtet at blive integreret ind i den gaussiske proces' kerne.

List of Publications

The following papers constitute contributions included in this thesis. See appendix A for attached copies of each.

1. R Bonnevie, M N Schmidt, and M Mørup (2017). "Difference-of-Convex optimization for variational kl-corrected inference in dirichlet process mixtures". In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6

2. R Bonnevie, M N Schmidt (2018). "Joint expectation kernels". Submitted to: 2018 Neural Information Processing Systems (NIPS).

3. R Bonnevie, M N Schmidt (2018). "Matrix Product states for inference in discrete probabilistic models". Submitted to: Journal of Machine Learning Research.

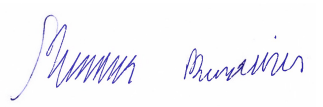
Preface

This PhD thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark, in partial fulfillment of the requirements for acquiring a PhD degree at the Technical University of Denmark.

This PhD project was financed by a DTU Compute PhD Scholarship and carried out at the ITMAN Graduate school at DTU Compute.

The PhD has involved the authoring of three papers, one published and two under review. The three articles are collected in the appendices of this thesis. The main chapters of the thesis will introduce the topics that these three papers are concerned with and note our contributions to each.

Lyngby, 01-July-2018



Rasmus Bonnevie

Acknowledgements

I have been extremely grateful for the opportunity these last three years to pursue a PhD degree at the Technical University of Denmark, courtesy of a scholarship from the institute. It has been a great chance to immerse myself in many of the exciting fields and to pursue a large variety of research projects; some fruitful, some not, but all exciting forays into the frontiers of machine learning.

I would like to thank my two supervisors Mikkel N. Schmidt and Morten Mørup who have always been there whenever needed, have always been extremely supportive and welcoming, and with whom I have had many great scientific discussions.

Also thanks to my two long-term (although only shortly overlapping) office companions Michael Riis Andersen and Peter Jørgensen for listening to all my fledgling ideas, for not feeling too pestered by my inquiries into their own work, and for allowing me to span most of the whiteboard.

More generally, I would like to thank all of my colleagues at the Cognitive Systems section for their company, the many great discussions we have had, their good spirits, and their helpfulness. The Cognitive Systems section has been a tremendous place to work, and I would like to thank Lars Kai Hansen, Wanja Andersen, and Sine Ingemann for making everything run so smoothly.

Finally, I would like to thank my parents Naja and Finn who have inadvertently set me on this path and helped keep my spirits high on the way, and my fiancée Ida who has supported me unwaveringly throughout my PhD, kept me company along the way, and who has patiently endured many explanations about Bayesian inference.

Notation

A small note on notation might be in order. We will otherwise try to define most conventions and definitions as they come up. As is customary, we will generally reserve unformatted characters like a, x, γ for scalars, and boldface characters like $\boldsymbol{x}$ and $\boldsymbol{v}$ for vectors. Bold will also be used to denote matrices, which will also often be upper case like $\boldsymbol{X}$. Random variables will be marked mostly with lower-case —out of convenience, to distinguish them from matrices, because of common practice, and because they will often be conflated with their instantiations when considering densities as functions— but we will occasionally use upper-case when we want to emphasize that it is a random variable. Finally, we use calligraphic fonts like $\mathcal{T}$ or $\mathcal{G}$ for tensors of higher order.

When indexing, we sometimes feel the need to use explicit notation $[\boldsymbol{X}]_{ij}$ to mean the element in the i'th row and j'th column of matrix $\boldsymbol{X}$, but most of the time we will just use X_{ij} or v_i. We will generally format the quantity according to the type of the indexed element, so v_i is the i'th scalar element of $\boldsymbol{v}$, but $\boldsymbol{v}_i$ is the i'th vector instance or subcomponent of $\boldsymbol{v}$. This comes from a need to occasionally conflate variables and sets of variables.

We will mostly use p and q for probability densities, and we will rarely feel the need to explicitly label them with their respective random variables, which will often be clear from their arguments, i.e. we write $p(x)$ and not $p_x(x)$. Expectations will be written in a compact manner without subscript if it is otherwise clear what the expectation is over and with respect to what distribution. When necessary we will use a density subscript to denote $\mathbb{E}_q[\cdot]$ to denote what distribution the expectation is with respect to, or a random variable subscript $\mathbb{E}_x[\cdot]$ to denote that the expectation is only over the marginal of that variable. If we feel the need to be very explicit, we write $\mathbb{E}_{q(x)}[\cdot]$. Very rarely we will also use sampling statements $x \sim q$, especially if the expectation is with respect to different densities for different variables. The conditioning bar $|$ will be used to denote conditional expectations.

Contents

Chapter 1

Introduction

When faced with difficult decisions in complex environments, we rarely have the luxury of a perfect model, and have to base our decisions on approximations of this ideal. Even limiting ourselves to theoretically viable models, there would still be too many possible models to search through, however, and even if there were not, we would be facing the problem of information deficiency: the more options we leave ourselves, the more information we need to collect to pick out the correct one. Making assumptions about the decision and the environment and the approximation we use to approximate either reduces the problem space and we can learn more about the system, within the scope of our approximation, with less information.

The concept of letting collected data inform our beliefs about unknown quantities is best quantified in terms of probability theory. As demonstrated in e.g. Jaynes (2003), the calculus of epistemic belief is mathematically equivalent to probability theory, making it the only sound foundation for inference and learning under the associated axioms. If $\mathcal{D}$ specifies our data, the observations we have made about the system we want to model, and $\boldsymbol{\theta}$ specifies the set of unknown quantities then the conditional probability $p(\mathcal{D}|\boldsymbol{\theta})$ specifies the probability of observing $\mathcal{D}$, given fixed $\boldsymbol{\theta}$. Implicitly, the distribution describes a random generative process for the $\mathcal{D}$, with the probability reflecting the likelihood that this hypothetical simulator recreates the observed data. Conversely, if a certain value of $\boldsymbol{\theta}$ has high probability of generating the observed data $\mathcal{D}$, this lends credence to the

belief of that value of $\boldsymbol{\theta}$ being correct. As a function of the unknown $\boldsymbol{\theta}$, we call this the likelihood, and the flexibility of the generative model encoded in it is the first in which we can add structure and implement assumptions.

The second method by which we can introduce prior knowledge and assumptions is the aptly named prior $p(\boldsymbol{\theta})$ on the unknown quantities. This distribution reflects our prior belief and assumptions about the scale, domain, and properties of each random quantity contained in $\boldsymbol{\theta}$, as well as any constraints it might observe. We might assume positivity for a scalar quantity, or we might restrict a matrix parameter to the manifold of orthogonal matrices.

To update our prior belief to take the evidence lent by the data into account, we can apply Bayes' theorem,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{1.1}$$

which gives us the so-called posterior $p(\boldsymbol{\theta}|\mathcal{D})$, our updated belief about the unknown quantities. The theorem is extremely important, but is actually a rather straightforward consequence of the probability axioms, being a simple reordering of the product relation $p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = p(\mathcal{D})p(\boldsymbol{\theta}|\mathcal{D})$. The proportionality statement from above is to emphasize that the shape of the posterior distribution is completely determined by the product of the likelihood and the prior; as such, the posterior is a sort of compromise, which will put high or low mass in a region if both the likelihood and the prior is high or low, respectively, and in the regions where they disagree it will level out to the degree that they cancel each other out. It is the relative value that counts, so it is the peaks and troughs that matter. Flat likelihoods imply that the observations lend little evidence to one value of $\boldsymbol{\theta}$ over another; this then means that we need a lot of observations before the likelihood starts to peak, and the posterior concentrates meaningfully. Flat priors similarly imply that we have no knowledge of what parameter values are more likely than another, and that we let the likelihood dominate. Some people advocate flat priors, believing that any type of informed prior introduces an unjustified subjective bias, but a more sensible perspective is to realize that any model — both the choice of likelihood and the choice of prior — directly constitute assumptions about the data-generating process, and that flat priors can sometimes translate into assumptions that are less justified than informed priors, such as assuming that the mean of a Gaussian is a priori equally likely to be 1 or one million.

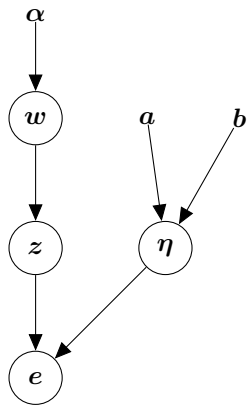


Figure 1.1: Simple Bayesian network of the stochastic block model.

1.1 Structure in Bayesian Probabilistic Models

It is possible to express many kinds of structure in a Bayesian model, but independence and conditional independence structure is amongst the most important, and many intuitive models naturally embody this type of structure.

An interesting case study illustrating conditional independence is the stochastic block model, which is a typical example of a hierarchical model exhibiting both discrete and continuous random variables. The setup is a partially observed undirected graph on N vertices, with a set $(i, j) \in E_{\text{obs}}$ cataloging the observed links, with (i, j) indicating an observation of the link between node i and j. We let e_{ij} be the corresponding binary variable revealing whether we observed an edge (1) or not (0). The stochastic block model tries to cluster the N vertices of the graph into K communities (clusters), and it can be defined as,

$$\boldsymbol{w} \sim \text{Dir}(\boldsymbol{\alpha}) \tag{1.2}$$

$$\eta_{kl} \sim \text{Beta}(a_{k\ell}, b_{k\ell}), \quad \forall j \in 1, \ldots, K,\ k \in 1, \ldots, \ell \tag{1.3}$$

$$z_i | \boldsymbol{w} \sim \text{Cat}(\boldsymbol{w}), \quad \forall i \in 1, \ldots, N \tag{1.4}$$

$$e_{ij} | \boldsymbol{\eta}, z_i, z_j \sim \text{Ber}(\eta_{z_i z_j}), \quad \forall (i, j) \in E_{\text{obs}}, \tag{1.5}$$

where we let $\eta_{z_i z_j}$ correspond to the $\eta_{k\ell}$ indexed by the clusters indicated by the categorical variables z_i and z_j, and we let bold indicate sets of variables. $\boldsymbol{\alpha}$ is a positive vector of length K and $a_{k\ell}$ and $b_{k\ell}$ are scalars, which together constitute the fixed hyperparameters of the model.

We can read the generative process prescribed by our model from this list: first, we pick the proportions of nodes belonging to each community $\boldsymbol{w}$, then we pick the strengths $\eta_{k\ell}$ of each possible cluster-to-cluster connection, and we assign nodes to clusters in the assigned proportions. Finally, we connect edges between two clusters k and ℓ relative to the strength $\eta_{k\ell}$ of that pairing. Written using conditional distributions, this gives us the following factorization of the joint distribution over all of the variables,

$$p(\boldsymbol{e}|\boldsymbol{z},\boldsymbol{\eta})p(\boldsymbol{z}|\boldsymbol{w})p(\eta|\boldsymbol{a},\boldsymbol{b})p(\boldsymbol{w}|\boldsymbol{\alpha}) \tag{1.6}$$

This factorization structure is what we mean by conditional independence structure. Using just the product relation from probability, we can always trivially write,

$$p(\boldsymbol{e}|\boldsymbol{z},\boldsymbol{\eta},\boldsymbol{w},\boldsymbol{a},\boldsymbol{b},\boldsymbol{\alpha})p(\boldsymbol{z}|\boldsymbol{\eta},\boldsymbol{w},\boldsymbol{a},\boldsymbol{b},\boldsymbol{\alpha})p(\boldsymbol{\eta}|,\boldsymbol{w},\boldsymbol{a},\boldsymbol{b},\boldsymbol{\alpha})p(\boldsymbol{w}|\boldsymbol{a},\boldsymbol{b},\boldsymbol{\alpha}) \tag{1.7}$$

no matter our assumptions, so stronger factorizations are a consequence of model assumptions. Another way to express the same assumptions is by way of a Bayesian network, a graphical model, where each node corresponds to one of the conditional probability factors of the factorized joint distribution, with the conditioning set determined by the parents of each node, as illustrated by the Bayesian network of the stochastic block model in figure 1.1.

It is worth mentioning that while the graphical model and factorization structure here follows the generative process, the product rule means that we can factor a joint model over N variables in $N!$ ways, so the factorization used here is neither unique nor necessarily causal. Essentially, looking at the graphical model, we can invert any of the arrows, at the possible cost of introducing new arrows to the graph. Since the factorization structure follows from the choice of model, there is often a particular factorization that is more natural, though.

When a distribution factorizes like this, it eases inference as we can distribute many operations across the factors, replacing a global computation with several local ones. Take marginalization as an example. It is a basic fact of probability theory that given the joint distribution $p(x_1, x_2, x_3)$ over three random variables, we can find the marginal distribution over just x_3 by integrating out, or marginalizing over, all possible values of x_1 and x_2

$$p(x_3) = \iint p(x_1, x_2, x_3)\,\mathrm{d}x_1\,\mathrm{d}x_2 \tag{1.8}$$

While this is of course a simple example, it is clear that we are facing a multi-dimensional integral over a potentially complicated function, and while a two-dimensional integral might be manageable it is clear that the curse of dimensionality would quickly come into play for even a modest number of random

variables, especially if we have to approximate said integrals. With conditional independence, this can be greatly simplified: if the model factorizes as $p(x_3)p(x_2|x_3)p(x_1|x_2)$, then the marginalization operation distributes like

$$\iint p(x_1)p(x_2|x_1)p(x_3|x_2)\,\mathrm{d}x_1\,\mathrm{d}x_2 = \int \left(\int p(x_1)p(x_2|p(x_1))\,\mathrm{d}x_1\right) p(x_3|x_2)\,\mathrm{d}x_2 \tag{1.9}$$

which boils down to a two-stage procedure of first computing

$$p(x_2) = \int p(x_1)p(x_2|p(x_1))\,\mathrm{d}x_1\,, \tag{1.10}$$

and then passing the result on to compute the final marginal

$$p(x_3) = \int p(x_3|x_2)p(x_2)\,\mathrm{d}x_2\,. \tag{1.11}$$

From a computational point of view, this improves on the original problem as we now only need to compute two univariate integrals — even if our model involved more random variables, we would still only have to compute univariate integrals, if the random variables possessed the same simple chain-like factorization structure. This basic premise of factorization structure allowing the distribution of computation is the key to most of the efficient inference algorithms today, from belief propagation and junction tree algorithms in exact discrete inference (Koller and Friedman 2009; Wainwright and Jordan 2008), over Kalman filters and Hidden Markov models (Chen 2003; Bishop 2006), to approximations capitalizing on the factorization structure such as Gibbs sampling and variational message passing, the latter of which we will discuss later (Minka 2005).

1.1.1 When Conditional Independence Breaks Down

Conditional independence is not a suitable assumption for all models, as random variables can exhibit mutual co-dependence and high covariance. A very simple example of this is a standard multivariate Gaussian with a dense precision (inverse covariance) matrix $\mathbf{\Sigma}^{-1}$. No matter how we try to factorize it, we cannot improve on the trivial factorization, as there are no conditional independencies to exploit. This is a peculiar feature of the Gaussian, that the sparsity pattern of the precision matrix matches the conditional independence structure exactly (Speed and Kiiveri 1986).

Gaussian processes are an apt example of this, as there is typically no way around working with unfactorized Gaussians. As Gaussian processes are effectively probability measures on function spaces, as we will talk more about in the chapter on Gaussian processes, this makes some intuitive sense as any one

instance of the random function will perfectly determine the value of all function evaluations simultaneously. Smoothness and continuity will likewise force the function to conform globally to the constraints imposed by local evaluations. This intuition can be misleading though; Matérn kernels in 1D encode varying degrees of smoothness and have dense and full-rank covariance matrices, but are actually Markov processes with an extreme conditional independence structure where each random variable only depends on the preceding value (Hartikainen and Särkkä 2010).

Finally, conditional independence can be compromised by marginalizing out auxiliary parameters. Take the stochastic block model from before; if we integrate out all the η_{kl} variables, we get the conditional distribution

$$p(\boldsymbol{e}|\boldsymbol{z}) = \prod_{k,\ell} \frac{B(a_{kl} + n_{kl}, b_{kl} + \bar{n}_{kl})}{B(a_{kl}, b_{kl})} \tag{1.12}$$

where n_{kl} and $\bar{n}_{kl}$ count the edges and non-edges, respectively, between clusters k and ℓ. Whereas the edges were generated independently previously, marginalizing out the hierarchical parameters $\eta_{k\ell}$ has led to a new likelihood where all the edges are generated jointly. This procedure also works in reverse sometimes; auxiliary variables can be introduced to represent complicated distributions in terms of larger distributions exhibiting conditional independencies; examples include general mixtures (Bishop 2006), the skew-normal distribution (Liseo and Loperfido 2003), and binomial distributions (Polson, Scott, and Windle 2013; Schein, Wallach, and Zhou 2016).

While marginalization often reduces the degree of factorization, it also offers advantages. For one, the marginal joint containing just the data and the variables of interest is the most compact description of the relationship between data and latent variables; auxiliary variables are by their very nature confounding, or at best irrelevant, elements to our analysis, and in an ideal setting we would perform our analyses without them. Reducing the dimensionality of the latent space is obviously a benefit in and of itself, and is particularly advantageous for any optimization-based inference procedure like MAP, both due to avoiding the problematic geometries of higher-dimensional spaces, but also as fewer tunable parameters could mean fewer local optima. Secondly, while the dependencies between the random variables are more intricate in the marginalized model, they are also more representative. With auxiliary variables, we might have that two variables of interest are weakly correlated under one configuration and strongly correlated under another. Once we marginalize, the joint influence of both these paradigms will be taken into account. Samplers have for instance been observed to navigate marginalized domains more efficiently (Teh, Newman, and Welling 2007).

1.1.2 Approximations and Dependencies

While samplers can move around collapsed spaces more adroitly, they come with their own shortcomings. The lack of reliable convergence diagnostics is often cited, but they also often scale poorly and can have trouble with discrete distributions. Variational methods, where we try to find an optimal surrogate for the true posterior, as we shall describe in more detail in the chapter on variational inference, tackle those issues, but face technical challenges during inference. The classical approaches rely intensely on conjugacy, exponential family distributions, and mean-field assumptions, which fits poorly with a marginalized model: marginalization of exponential family distributions almost universally results in densities outside of that class, and mean-field works best when random variables are only weakly-dependent.

More modern approaches to variational inference have tried to circumvent these issues, by making the inference methods more universally applicable and black-box. The central task is then arguably to find models that can adequately express the dependencies of the true posterior.

1.2 Papers and Thesis Overview

In this thesis and the accompanying papers, we will target the type of model detailed above where conditional independence structure is not available to exploit due to massive mutual interdependence of the random variables of interest. In particular, we will circle around partition-based models, such as mixtures and graph clustering models, and Gaussian processes, both of which exhibit this type of interdependence in a very natural way as briefly described above.

The thesis ties together the three papers authored as part of this PhD. The papers, and their connection to the main text, are described briefly below, with the chapters of the mean text followed by more detailed contribution sections.

Difference-of-convex optimization for variational KL-corrected inference in Dirichlet process mixtures by Rasmus Bonnevie, Morten Mørup, and Mikkel N. Schmidt, published as

> R Bonnevie, M N Schmidt, and M Mørup (2017). "Difference-of-Convex optimization for variational kl-corrected inference in dirichlet

process mixtures". In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6

The paper is concerned with inferring the cluster indicators of a Dirichlet process mixture model, leading to a partition-based model similar to the graph community model we described earlier once the continuous variables are marginalized out. Seeking a variational approximation, the paper uses conjugacy to arrive at the classical solution to the problem, and then the KL-corrected variational bound to approximately marginalize out the nuisance parameters. The central contribution is the discovery that the resulting bound has a difference-of-convex property amenable to structure-exploiting optimization algorithms. The resulting update steps turn out to be analytical, and we find a novel fixed-point formula. This result is diminished by finding that the fixed-point procedure matches the classical coordinate-ascent algorithm for variational Bayes exactly.

Joint Expectation Kernels by Rasmus Bonnevie and Mikkel N. Schmidt. Submitted to NIPS 2018. This paper explores the application of kernel methods to spaces of covariant random variables, with the original intent of finding ways to apply Bayesian quadrature to functions evaluated at unknown locations, as well as finding a more efficient method for Gaussian process regression with noisy input. This paper deviates from the full Bayesian approach that we otherwise advocate by proposing a kernel space embedding of the lower level of the probabilistic hierarchy, e.g. the noisy input points. This allows the direct application of kernel methods, and Gaussian processes in particular, to sets of covariant random variables. We contrast the proposed kernel with the existing kernel mean embedding methodology, noting conceptual and practical differences. Application examples include censored data and the embedding of entire Gaussian processes.

Matrix product states for inference in discrete probabilistic models by Rasmus Bonnevie and Mikkel N. Schmidt. Submitted to JMLR. The final paper tackles the previously detailed problem head-on by confronting the issue that most variational approximations to discrete distributions fail to take mutual dependency into account in any way, often using complete mean-field approximations where all random variables are assumed independent. For partition-based models we conjecture that this can potentially be very harmful, as it will prevent detection of alternative modes and obfuscate the true uncertainty. In this paper we argue that matrix product states generalize the existing mean-field methods while modeling dependencies much more explicitly. We also show how many probabilistic operations such as marginalization and conditioning can be carried

out efficiently, and how inference can be performed using Monte Carlo gradient estimators.

1.2.1 Structure of the Thesis

The thesis chapters are written with the intention of providing background on the topics covered in the papers, as well as illustrating and showcasing novel perspectives and recent innovations within each topic. We have divided the material into three distinct topics, with some overlapping content, and we briefly detail how each chapter relates back to one or more of the attached papers,

Chapter 1 on Variational Inference details an approximate inference methodology that has become pivotal over the last decade, namely that of variational inference, where the posterior is sought approximated by minimizing a divergence with respect to a surrogate. Chapter 1 will focus on the many developments over the last few years that has allowed the methodology to be extended to models of arbitrary complexity, and contrasts it with the classical approach based on conjugate models. This chapter relates to papers *Difference-of-convex optimization for variational KL-corrected inference in Dirichlet process mixtures* and *Matrix Product states for inference in discrete probabilistic models*, where the former embodies the classical approach and contrasts with the latter that tries to apply the modern innovations in the field.

Chapter 2 on Gaussian Processes covers the basics, as well as details on what assumptions are encoded in the choice of covariance kernel. We also describe how to extend the Gaussian process to larger Gaussian ensembles. For continuity, we will also explain how variational inference can be applied to Gaussian processes. This chapter relates to the *Joint expectation kernels* which describes a novel kernel that can be applied to spaces of covariant random variables.

Chapter 3 on Tensor Networks details a novel type of model from quantum mechanics and tensor factorization which can be used to model groups of highly interdependent discrete random variables. We focus on its relationship with probabilistic graphical models. The paper *Matrix Product states for inference in discrete probabilistic models* covers this topic in greater detail.

Chapter 2

Variational Inference

Variational inference (Jordan et al. 1999) has quickly become the workhorse of approximate Bayesian inference. While Markov Chain Monte Carlo (MCMC) (Bishop 2006; Gelman et al. 2014) continues to see widespread use in problems where accurate uncertainty quantification is essential, it is often charged with suffering from poor diagnostics, slow convergence, and poor mixing. While there has been significant progress on all of these points (Hoffman and Gelman 2014; Nemeth et al. 2017), and probabilistic programming languages like Stan have made MCMC a viable and dependable tool for the general research community (Carpenter et al. 2017), scalability continues to be a real issue for sampling-based algorithms. Aside from being convenient and occasionally elegant, scalability is really the main selling point of variational inference.

2.1 KL Divergence and the Lower Bound

The name of variational inference references the calculus of variations which is the study of optimization problems over spaces of functions. The central premise of variational inference is that if we have a divergence measure between distributions $D(\cdot\|\cdot)$, then we can formulate inference as the optimization problem,

$$p(z|x) \approx \underset{q \in \mathcal{F}}{\operatorname{argmin}}\, D(q(z), p(z|x)), \tag{2.1}$$

for a posterior $p(z|x)$ and some class of probability density functions $\mathcal{F}$, where we use x to denote a set of observed quantities (the "data") and where z is the latent quantity of interest. In other words, an approximation to the posterior is found by minimizing the divergence between an approximate surrogate and the true posterior. While the above expression describes the problem in all generality, variational inference has largely become synonymous with minimizing the reverse Kullback-Leibler divergence (Kullback and Leibler 1951; Minka 2005),

$$\mathrm{KL}\left(q(z) \| p(z|x)\right) = \mathbb{E}_{q(z)}\left[\ln \frac{q(z)}{p(z|x)}\right]. \tag{2.2}$$

The Kullback-Leibler divergence is notably not a distance in the metric sense as it is neither symmetric nor obeys the triangle inequality, but it does possess the two other properties of a metric, namely positivity $\mathrm{KL}\left(\cdot \| \cdot\right) \geq 0$ and discernibility $\mathrm{KL}\left(p \| q\right) = 0 \Leftrightarrow p = q$. It is also only well-defined if the p is absolutely continuous with respect to q for $\mathrm{KL}\left(p \| q\right)$, making it challenging to apply to degenerate distributions. It makes up for these lacks by being a Bregman divergence and intimately tied to information geometry (Amari and Nagaoka 2007), and also for generally being more amenable to analysis and application than more general f-divergences and α-divergences. It also has the quality of being zero-forcing, encouraging the approximate posterior to not put mass in low-density regions (Minka 2005). This last property means that when the approximation does not have the capacity to fit the true model, it would rather over-estimate the mode rather than the tails of the true posterior, leading to the often cited claims of underestimation of variance. As a result of its general tractability, the divergence also breaks down neatly into well-known components from information theory,

$$\mathrm{KL}\left(p(z) \| q(z)\right) = \underbrace{\left(-\,\mathbb{E}_{q(z)}[\ln p(z)]\right)}_{\text{cross-entropy}} - \underbrace{\left(-\,\mathbb{E}_{q(z)}[\ln q(z)]\right)}_{\text{entropy}} \tag{2.3}$$

A central issue with the divergence in equation (2.2) is that it requires prior knowledge of $p(z|x)$ to even evaluate. Getting around this turns out to be fairly straightforward, if we start by defining the so-called Evidence Lower Bound.

Definition 2.1 The Evidence Lower Bound (ELBO) for a distribution $p(x, z)$ is

$$\mathcal{L}(q) = \mathbb{E}_{q(z)}[\ln p(x, z)] - \mathbb{E}_{q(z)}[\ln q(z)] \tag{2.4}$$

The definition of the ELBO follows from a particular relationship existing between the KL divergence, the ELBO and the evidence $\ln p(x)$.

Lemma 2.2 *For a distribution $p(x, z)$ with ELBO $\mathcal{L}(q)$ the following relationship holds,*

$$\mathcal{L}(q) = \ln p(x) - \mathrm{KL}\left(q(z) \| p(z|x)\right). \tag{2.5}$$

Proof. We start with the KL divergence and multiply and divide by the evidence in the fraction.

$$\mathrm{KL}\left(q(z) \| p(z|x)\right) = \int q(z) \ln \frac{q(z)}{p(z|x)} \, \mathrm{d}z = \int q(z) \ln \frac{q(z)p(x)}{p(z, x)} \, \mathrm{d}z \,.$$

We can now pull out the evidence as it's constant under the integral. Rearranging then gives us the result. □

The above result explains why the ELBO is a lower bound on the evidence: since the divergence is positive, it must hold that $\mathcal{L}(q) \leq \ln p(x)$. In fact, since $\ln p(x)$ is constant, the above equality describes a much more powerful relationship between the divergence and the bound as they are only ever off by a constant. This is of particular interest as it implies that the ELBO and the KL-divergence share gradients, critical points, and optima — as such we lose nothing by substituting one for the other in an optimization problem. Due to this, the variational inference problem of equation (2.1) using a Kullback-Leibler divergence is often presented simply as,

$$\min_q \mathcal{L}(q). \tag{2.6}$$

Similarly to the way we could decompose the proper KL-divergence, we can also decompose the ELBO in at least two informative ways:

Prior-likelihood decomposition By expanding the joint distribution, the bound can be restated in terms of expected likelihood and the KL divergence between approximation and prior.

$$\mathcal{L}(q) = \mathbb{E}_q[\ln p(x|z)] - \mathrm{KL}\left(q(z) \| p(z)\right). \tag{2.7}$$

The first term is then clearly a data-fit term, while the latter constrains the optimal q to be close to the prior in the KL sense.

Energy-entropy decomposition If we view the joint as a Gibbs distribution $p(x, z) = e^{E(x,z)}$, then we can write

$$\mathcal{L}(q) = \mathbb{E}[E(x, z)] + \mathcal{H}(q), \tag{2.8}$$

where $\mathcal{H}(q) = -\mathbb{E}_{q(z)}[\ln q(z)]$ is the differential Shannon entropy. The entropy is the interesting part here, as maximum entropy is a common principle for determining non-informative distributions.

In both of these decompositions, we see a trade-off between q allocating probability mass to intervals with a high likelihood, and a regularizing effect: either due to minimizing the divergence from the prior, or due to a maximization of the entropy of the approximation.

Tractability now comes down to whether the two expectations can be computed. We will contrast two paradigms: the case of conjugate exponential family mean-field models where everything is tractable, and the more recent black-box setup which can be applied to any model.

2.2 Gradients and Local Updates

Variational inference boils down to solving an optimization problem, which is both a key advantage, but also a significant challenge. Most optimization algorithms use gradients to change parameters incrementally in a gradient ascent procedure, but we can also consider solving the complete problem in an iterative fashion by solving a sequence of tractable subproblems. We will detail these two variational inference designs below, starting with the more generally applicable method.

2.2.1 Gradient Estimators

The most general optimization strategy for variational inference is also one of the most recent. For expectations over distributions $q(\boldsymbol{z};\boldsymbol{\eta})$ that are differentiably parameterized with parameters $\boldsymbol{\eta}$, it turns out we can take derivatives through the expectation operator using the simple result (Ranganath, Gerrish, and Blei 2013),

Lemma 2.3 *For a function $f(\boldsymbol{z},\boldsymbol{\eta})$, differentiable in $\boldsymbol{\eta}$, and a distribution $q(\boldsymbol{z};\boldsymbol{\eta})$ which is likewise differentiable in $\boldsymbol{\eta}$, the following result holds*

$$\boldsymbol{\nabla}_{\boldsymbol{\eta}}\,\mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\eta})}[f(\boldsymbol{z},\boldsymbol{\eta})] = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\eta})}[(\boldsymbol{\nabla}_{\boldsymbol{\eta}}\ln q(\boldsymbol{z};\boldsymbol{\eta}))\,f(\boldsymbol{z},\boldsymbol{\eta}) + \boldsymbol{\nabla}_{\boldsymbol{\eta}} f(\boldsymbol{z},\boldsymbol{\eta})]\,. \tag{2.9}$$

Proof. Expanding the expectation integral and moving the gradient inside, which is allowed under some weak regularity conditions discussed in Ranganath, Gerrish, and Blei (2013), we can apply the product rule

$$\boldsymbol{\nabla}_{\boldsymbol{\eta}} \int q(\boldsymbol{z};\boldsymbol{\eta}) f(\boldsymbol{z},\boldsymbol{\eta})\,\mathrm{d}\boldsymbol{z} = \int (\boldsymbol{\nabla}_{\boldsymbol{\eta}} q(\boldsymbol{z};\boldsymbol{\eta}))\,f(\boldsymbol{z},\boldsymbol{\eta}) + q(\boldsymbol{z};\boldsymbol{\eta})\boldsymbol{\nabla}_{\boldsymbol{\eta}} f(\boldsymbol{z},\boldsymbol{\eta})\,\mathrm{d}\boldsymbol{z}\,. \tag{2.10}$$

Now, we can simply apply the log-derivative trick $\boldsymbol{\nabla}_{\boldsymbol{\eta}} q(\boldsymbol{z};\boldsymbol{\eta}) = q(\boldsymbol{z};\boldsymbol{\eta})\boldsymbol{\nabla}_{\boldsymbol{\eta}} \ln q(\boldsymbol{z};\boldsymbol{\eta})$ to reintroduce $q(\boldsymbol{z};\boldsymbol{\eta})$ as a factor, allowing us to write the whole thing as an expectation, proving the relation. □

With the above result in hand, it's straightforward to rewrite the gradient of the ELBO as

$$\boldsymbol{\nabla}_{\boldsymbol{\eta}} \mathcal{L}(q(\boldsymbol{z};\boldsymbol{\eta})) = \mathbb{E}_q\left[(\boldsymbol{\nabla}_{\boldsymbol{\eta}} \ln q(\boldsymbol{z};\boldsymbol{\eta})) \ln \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z};\boldsymbol{\eta})} - \boldsymbol{\nabla}_{\boldsymbol{\eta}} \ln q(\boldsymbol{x};\boldsymbol{\eta})\right]. \tag{2.11}$$

A further simplification is possible as the last term, the so-called score function $\boldsymbol{\nabla}_{\boldsymbol{\eta}} \ln q(\boldsymbol{x};\boldsymbol{\eta})$ of q, has expectation 0 under q. Having moved the derivative operator inside the expectation, we can now use Monte Carlo to construct unbiased estimators of the gradients:

$$\nabla_{\boldsymbol{\eta}} \mathcal{L}(q(\boldsymbol{z};\boldsymbol{\eta})) \approx \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\nabla}_{\boldsymbol{\eta}} \ln q(\boldsymbol{x};\boldsymbol{\eta}) \ln \frac{p(\boldsymbol{x},\boldsymbol{z}_k)}{q(\boldsymbol{z}_k;\boldsymbol{\eta})}, \quad \boldsymbol{z}_k \sim q(\boldsymbol{z};\boldsymbol{\eta}) \tag{2.12}$$

Plugging this estimator directly into a stochastic gradient algorithm, we can perform stochastic optimization on the ELBO. This particular estimator is known variously as the black box estimator, the score function estimator, or the REINFORCE estimator (R. J. Williams 1992; Ranganath, Gerrish, and Blei 2013). This general idea is arguably what has led to the current popularity of variational inference, as it can be applied to arbitrary differentiable models. The recurring problem is that although the estimator is unbiased, it can have tremendously high variance. Soon after the above estimator was proposed, another estimator grew into prominence (Salimans and Knowles 2013; Kingma and Welling 2013; Rezende and Mohamed 2014), building on the simple Law of the Unconscious Statistician,

$$x \overset{(D)}{=} h(\epsilon) \Rightarrow \mathbb{E}_x[f(x)] = \mathbb{E}_\epsilon[f(h(\epsilon))] \tag{2.13}$$

where $\overset{(D)}{=}$ denotes equality in distribution between two random variables. The so-called reparametrization estimator constructs a gradient estimator simply by moving the parameter out of the expectation, and then moving the gradient operator inside the expectation,

$$\boldsymbol{\nabla}_{\boldsymbol{\eta}} \mathbb{E}_{q(\boldsymbol{z})}[f(\boldsymbol{z},\boldsymbol{\eta})] = \mathbb{E}_{q_0(\boldsymbol{\epsilon})}[\boldsymbol{\nabla}_{\boldsymbol{\eta}} f(h(\boldsymbol{\epsilon};\boldsymbol{\eta}),\boldsymbol{\eta})], \quad \boldsymbol{z} \overset{(D)}{=} h(\boldsymbol{\epsilon};\boldsymbol{\eta}), \quad \boldsymbol{\epsilon} \sim q_0(\boldsymbol{\epsilon}). \tag{2.14}$$

Empirically, this has much lower variance, intuitively since we get to factor in the dependence of the objective function f and apply the chain rule, as opposed to the product rule used for the black-box estimator.

The problem with reparameterization is that we need to find a new transform for every distribution. All distributions with tractable CDFs are easy, as we can simply use the inverse transform sampler to map uniform samples to the density in question (Bishop 2006). A lot of solutions have been proposed for other distributions, such as differentiable rejection sampling (Naesseth et al. 2017) or generalized reparameterizations that allow noise distributions to depend weakly on the parameters (Ruiz, Titsias, and Blei 2016b).

2.2.1.1 Variance Reduction

The key to making gradient estimators work is to ensure the variance is sufficiently small. Trivially, if it was zero and we had access to the true gradient we would inherit all the guarantees of standard gradient descent, and while stochastic gradient descent does guarantee asymptotic convergence for unbiased gradients, high variance pushes the actual point of convergence further towards the asymptotic regime (Robbins and Monro 1951). Various methods for reducing the variance of Monte Carlo estimates have existed for decades in optimization and statistics, but the initial papers proposing black-box gradient estimators were quick to point out the importance of the topic (Ranganath, Gerrish, and Blei 2013; Kingma and Welling 2013).

One strategy for variance reduction which is well-known in the statistics community, albeit for other qualities, is importance sampling. The general conceit is to estimate expectations by sampling from one distribution, and then weighting the samples before averaging to turn the sample estimator into an estimator for a different distribution. Mathematically,

$$\mathbb{E}_q[f(\boldsymbol{z})] = \int q(\boldsymbol{z}) f(\boldsymbol{z}) \,\mathrm{d}\boldsymbol{z} = \int r(\boldsymbol{z}) \frac{q(\boldsymbol{z})}{r(\boldsymbol{z})} f(\boldsymbol{z}) \,\mathrm{d}\boldsymbol{z} = \mathbb{E}_r\left[\frac{q(\boldsymbol{z})}{r(\boldsymbol{z})} f(\boldsymbol{z})\right], \tag{2.15}$$

where we name r the proposal distribution, which we get to pick arbitrarily. In most applications, importance sampling is simply used to generate samples from a complicated q distribution, but we can start to speculate in whether some choices of r lead to better estimators, specifically ones with lower variance. We can actually find the optimal unnormalized proposal quite easily by applying the calculus of variations to minimize the variance

$$\int r(\boldsymbol{z}) \left(\frac{q(\boldsymbol{z})}{r(\boldsymbol{z})} f(\boldsymbol{z}) - \bar{f}\right)^2 \mathrm{d}\boldsymbol{z} \tag{2.16}$$

with respect to the proposal r, where $\bar{f} = \mathbb{E}_q[f(\boldsymbol{z})]$. Disregarding the arguments for notational convenience, and rewriting the variance expression with a Lagrange

multiplier and an exponential to ensure positivity,

$$\int e^r \left(\frac{q}{e^r} f - \bar{f}\right)^2 \mathrm{d}\boldsymbol{z} + \lambda \int e^r \,\mathrm{d}\boldsymbol{z} - \lambda, \tag{2.17}$$

we see that the derivative of the integrand with respect to r yields,

$$e^r \left(\frac{q}{e^r} f - \bar{f}\right)^2 - 2e^r \left(\frac{q}{e^r} f - \bar{f}\right) \frac{q}{e^r} f + \lambda e^r. \tag{2.18}$$

Setting this to zero, we can isolate e^r after some algebra,

$$e^r = \frac{q|f|}{\sqrt{\bar{f}^2 + \lambda}} \propto q|f|. \tag{2.19}$$

This is the minimal variance proposal, but ironically normalizing it requires us to solve the original expectation problem (at least if f is positive). If we plug it into the importance sampling relation of equation (2.15) and estimate the expectation with a single sample $\hat{\boldsymbol{z}} \sim r$, it reduces to

$$\frac{q(\hat{\boldsymbol{z}}) f(\hat{\boldsymbol{z}})}{\frac{1}{\mathbb{E}_q[|f(\boldsymbol{z})|]} q(\hat{\boldsymbol{z}}) |f(\hat{\boldsymbol{z}})|} = \operatorname{sign}(f(\hat{\boldsymbol{z}}))\, \mathbb{E}_q[|f(\boldsymbol{z})|] \tag{2.20}$$

which, if f is positive, means that the estimator has 0 variance as the estimator is just the desired expectation. This optimal density is obviously out of reach in any practical application, but approximating it should bring down the variance. This intuition is behind the overdispersed black-box variational inference procedure, which argues that $q|f|$ tends to have heavier tails than q, and thus modifying q to have higher variance and broader tails reduces variance (Ruiz, Titsias, and Blei 2016a).

Another popular strategy is control variates, which was brought up in the context of gradient estimators for VI already during the original presentation of the black-box method (Ranganath, Gerrish, and Blei 2013). The slightly counterintuitive idea is to reduce variance by adding more randomness in the form of a zero-mean random variable, with the saving grace that it is negatively correlated with the target variable. In its simplest form we have

$$\mathbb{E}_q[f(\boldsymbol{z})] = \mathbb{E}\Big[f(\boldsymbol{z}) + \alpha \hat{h}\Big] \tag{2.21}$$

where $\hat{h}$ is the arbitrary random variable we call the control variate, and $\alpha \in \mathbb{R}$ is a weight. Again, we can also write the expression for the variance,

$$\operatorname{Var}\Big(f(\boldsymbol{z}) + \alpha \hat{h}\Big) = \operatorname{Var}(f(\boldsymbol{z})) + \alpha^2 \operatorname{Var}\Big(\hat{h}\Big) + 2\alpha \operatorname{Cov}\Big(\hat{h}, f(\boldsymbol{z})\Big). \tag{2.22}$$

If we take the derivative with respect to α and set it to 0, we find the following optimal value,

$$\alpha = -\frac{\mathrm{Cov}\Big(\hat{h}, f(\boldsymbol{z})\Big)}{\mathrm{Var}\Big(\hat{h}\Big)}. \tag{2.23}$$

Plugging it back into the variance expression, we get that ,

$$\mathrm{Var}\Big(f(\boldsymbol{z}) + \alpha\hat{h}\Big) = \mathrm{Var}(f(\boldsymbol{z})) - \frac{\mathrm{Cov}\Big(\hat{h}, f(\boldsymbol{z})\Big)^2}{\mathrm{Var}\Big(\hat{h}\Big)}, \tag{2.24}$$

which implies that the optimal choice of $\hat{h}$ is one that maximizes the covariance with $f(\boldsymbol{z})$. On a first look, the best choice would be $f(\boldsymbol{z})$, but we have skipped a bit lightly over the constraint that the control variate should be zero mean, so the best choice is actually $f(\boldsymbol{z}) - \mathbb{E}[f(\boldsymbol{z})]$ which leads to the same kind of circular reasoning we saw for the importance sampling scheme.

Fulfilling zero-mean and high covariance simultaneously is actually a challenging design problem as the closer we get to $f(\boldsymbol{z})$, the more the control variate inherits the expectation difficulties of the original problem. Sometimes a convenient candidate presents itself, like with the black-box estimator which has the integrand

$$\nabla_{\boldsymbol{\eta}} \ln q(\boldsymbol{z};\boldsymbol{\eta}) \ln \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z};\boldsymbol{\eta})} - \nabla_{\boldsymbol{\eta}} \ln q(\boldsymbol{z};\boldsymbol{\eta}). \tag{2.25}$$

It seems likely that this should have high covariance with $\nabla_{\boldsymbol{\eta}} \ln q(\boldsymbol{z};\boldsymbol{\eta})$ which is the so-called score function of q, which conveniently happens to always have expectation 0 with respect to its underlying distribution (under some weak regularity conditions). This makes the score readily applicable as a control variate, and it was proposed alongside the black-box estimator (Ranganath, Gerrish, and Blei 2013).

A more recent and general purpose design is based on Stein's identity which states that (Stein 1972; Liu, Lee, and Jordan 2016),

Lemma 2.4 (Stein's Identity) *For a smooth density $q(\boldsymbol{z})$, it holds that*

$$\mathbb{E}\big[(\nabla_{\boldsymbol{z}} \ln q(\boldsymbol{z};\boldsymbol{\eta}))\boldsymbol{\phi}(\boldsymbol{z})^\top + \nabla_{\boldsymbol{z}}\boldsymbol{\phi}(\boldsymbol{z})\big] = \mathbf{0}, \tag{2.26}$$

if $\int \nabla_{\boldsymbol{z}}(q(\boldsymbol{z})\boldsymbol{\phi}(\boldsymbol{z}))\,\mathrm{d}\boldsymbol{z} = \mathbf{0}$.

The conditions for Stein's identity are automatically fulfilled for $\boldsymbol{\phi}$ going to 0 on the boundary and bounded $q(\boldsymbol{z})$ (Liu, Lee, and Jordan 2016). This is readily

transformed into a control variate by assuming $\phi : \mathbb{R}^D \to \mathbb{R}$ and summing over the elements leaving,

$$\hat{h} = (\nabla_{\boldsymbol{z}} \cdot \ln q(\boldsymbol{z}; \boldsymbol{\eta}))\phi(\boldsymbol{z}) + \nabla_{\boldsymbol{z}} \cdot \phi(\boldsymbol{z}) \tag{2.27}$$

where $\nabla\cdot$ is the divergence operator. This design has been used successfully in conjunction with kernel methods (Oates, Girolami, and Chopin 2017) and a neural version was also recently proposed (Zhu, Wan, and Zhong 2018).

A version of control variates that will become especially relevant to us later is the REBAR/RELAX method (Tucker et al. 2017; Grathwohl et al. 2017). For this, we assume there is an auxiliary random variable $\boldsymbol{u}$ and a function $\hat{f}$, so that we can write

$$\mathbb{E}[f(\boldsymbol{z})] = \mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{z})] - \alpha\, \mathbb{E}_{\boldsymbol{u}}\left[\hat{f}(\boldsymbol{u})\right] + \alpha\, \mathbb{E}_{\boldsymbol{u}}\left[\hat{f}(\boldsymbol{u})\right] = \tag{2.28}$$

$$\mathbb{E}_{\boldsymbol{z}}\left[f(\boldsymbol{z}) - \alpha\, \mathbb{E}_{\boldsymbol{u}|\boldsymbol{z}}\left[\hat{f}(\boldsymbol{u})\right]\right] + \alpha\, \mathbb{E}_{\boldsymbol{u}}\left[\hat{f}(\boldsymbol{u})\right] \tag{2.29}$$

where we have used the tower identity $\mathbb{E}_{\boldsymbol{u}}[\cdot] = \mathbb{E}_{\boldsymbol{z}}\left[\mathbb{E}_{\boldsymbol{u}|\boldsymbol{z}}[\cdot]\right]$. This type of control variate is particularly convenient when $q(\boldsymbol{z})$ is not reparameterizable, but $q(\boldsymbol{u})$ and $q(\boldsymbol{u}|\boldsymbol{z})$ are. This becomes apparent if we take the gradient of equation (2.29),

$$\mathbb{E}_{\boldsymbol{z}}\left[\left(f(\boldsymbol{z}) - \alpha\, \mathbb{E}_{\boldsymbol{u}|\boldsymbol{z}}\left[\hat{f}(\boldsymbol{u})\right]\right) \nabla \ln q(\boldsymbol{z}; \boldsymbol{\eta}) - \alpha \nabla\, \mathbb{E}_{\boldsymbol{u}|\boldsymbol{z}}\left[\hat{f}(\boldsymbol{u})\right]\right] + \alpha \nabla\, \mathbb{E}_{\boldsymbol{u}}\left[\hat{f}(\boldsymbol{u})\right]. \tag{2.30}$$

The original use case for the estimator was models with discrete $\boldsymbol{z}$. It is impossible to find a differentiable reparameterization of a discrete variable, but we can often find a differentiably reparameterizable variable $\boldsymbol{u}$ such that $\boldsymbol{z} = H(\boldsymbol{u})$ where H is a non-differentiable map. Assuming that $\boldsymbol{u}|\boldsymbol{z}$ is tractable and reparameterizable, this falls into the above category. Both binary and categorical random variables can be reparameterized in this manner (Tucker et al. 2017). We extend it to matrix product state models in our own work by adopting a strategy for binary neural networks.

For completeness, we should also mention Rao-Blackwellization (Ranganath, Gerrish, and Blei 2013). Rao-Blackwell's theorem is a statement about the variance of estimators, a consequence of which is that for an arbitrary estimator $g(X, Y)$ depending on two random variables X and Y,

$$\mathrm{Var}_X(\mathbb{E}[g(X, Y)|X]) \leq \mathrm{Var}_{X,Y}(g(X, Y)). \tag{2.31}$$

where $\mathbb{E}[\cdot|X]$ is the conditional expectation. Intuitively, integrating out randomness reduces variance. The most successful method in this paradigm is the

local expectation gradient (Titsias and Lázaro-Gredilla 2015), which can be summarized with the relation

$$\nabla_{\boldsymbol{\eta}} \mathbb{E}[f(\boldsymbol{z})] = \mathbb{E}_{\boldsymbol{z}_{-i}}\big[\mathbb{E}_{z_i|\boldsymbol{z}_{-i}}[f(\boldsymbol{z}_{-i}, z_i)\nabla_{\boldsymbol{\eta}} \ln q(\boldsymbol{z}_{-i}, z_i; \boldsymbol{\eta})]\big] \tag{2.32}$$

where we have split $\boldsymbol{z}$ into its i'th element z_i and the remaining elements $\boldsymbol{z}_{-i}$. Note the expectation is also split into a multivariate and univariate part. If we can solve the latter, Rao-Blackwell guarantees lower variance. For discrete problems, this can be quite straightforward. We can further pick different z_i to marginalize over for each element of the gradient; to prevent superfluous sampling we can sample a single pivot sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}; \boldsymbol{\eta})$ and use it for all of the outer expectations, as $\boldsymbol{z}^*_{-i} \sim q(\boldsymbol{z}_{-i}; \boldsymbol{\eta})$.

2.2.2 Local Updates

Assuming that sets of latent variables are independent in the posterior will weaken the approximation, but can pay out in terms of increased tractability. If the latent quantity $\boldsymbol{z}$ breaks down into a set of latent variables or subsets of latent variables $\boldsymbol{z} = \{\boldsymbol{z}_i\}_{i=1}^N$, we can assume that the sets are independent under our approximation

$$q(\boldsymbol{z}) = \prod_{i=1}^{N} q(\boldsymbol{z}_i). \tag{2.33}$$

Applying this assumption to the ELBO, we can decompose it into parts

$$\mathcal{L}(q) = \mathbb{E}_{q(\boldsymbol{z}_i)}[\mathbb{E}_{\boldsymbol{z}_i}[\ln p(\boldsymbol{z}, \boldsymbol{x})]] - \mathbb{E}_{\boldsymbol{z}_i}[\ln q(\boldsymbol{z}_i)] + \text{const.} = \tag{2.34}$$

$$\mathbb{E}_{q(\boldsymbol{z}_i)}\left[\ln \frac{1}{\mathcal{Z}} \exp\big(\mathbb{E}_{\boldsymbol{z}_{/i}}[\ln p(\boldsymbol{z}, \boldsymbol{x})]\big)\right] - \mathbb{E}_{\boldsymbol{z}_i}[\ln q(\boldsymbol{z}_i)] + \text{const.} \tag{2.35}$$

where we introduced the constant normalization constant $\mathcal{Z}$. The two first terms together form a negative KL divergence, so with respect to $q(\boldsymbol{z}_i)$ alone, the bound is maximized by matching the two elements in the divergence, i.e.

$$q^*(\boldsymbol{z}_i) = \frac{1}{\mathcal{Z}} \exp\big(\mathbb{E}_{\boldsymbol{z}_{/i}}[\ln p(\boldsymbol{z}, \boldsymbol{x})]\big). \tag{2.36}$$

Note that this result follows without making any parametric assumptions; as with Bayes' theorem, the functional form of the solution follows directly. This sequential update scheme was once synonymous with variational Bayes, but to contrast it with other optimization strategies it is often referred to as the coordinate-ascent update, as it corresponds to an optimal application of that algorithm.

2.2.3 Variational Message Passing

In many scenarios, our model will factorize according to some Bayesian network describing conditional independence relationships, letting us write

$$p(\boldsymbol{z}) = \prod_{i=1}^{N} p(\boldsymbol{z}_i|\mathrm{pa}_i) \tag{2.37}$$

where pa_i is the set of parents of $\boldsymbol{z}_i$ in the Bayesian network graph, and we have subsumed the observed $\boldsymbol{x}$ into the set of random variables $\boldsymbol{z}$ for notational simplicity. Substituting this into the logarithm of equation (2.36), we get

$$\ln q^*(\boldsymbol{z}_i) = \sum_{j=1}^{N} \mathbb{E}\big[\ln p(\boldsymbol{z}_j|\mathrm{pa}_j)|\boldsymbol{z}_i, \boldsymbol{x}\big] + \text{const.} \tag{2.38}$$

Now, if $j \neq i$ and $i \notin \mathrm{pa}_j$ the j'th term will reduce to a constant. If $i \in \mathrm{pa}_j$, we get the symmetric relationship $j \in \mathrm{ch}_i$, with ch_i denoting the set of children nodes of i. Since the expectation is then just over the children and parents of $\boldsymbol{z}_i$ (ch_i and pa_i), and the parents of the children pa_j for $j \in \mathrm{ch}_i$, the expectation is exactly restricted to the Markov blanket mb_i which is defined to be this exact set of variables. So the equation simplifies to (Winn and Bishop 2005)

$$\ln q^*(\boldsymbol{z}_i) = \mathbb{E}[\ln p(\boldsymbol{z}_i|\mathrm{pa}_i)|\boldsymbol{z}_i, \boldsymbol{x}] + \sum_{j\in\mathrm{ch}_i} \mathbb{E}\big[\ln p(\boldsymbol{z}_j|\mathrm{pa}_j)|\boldsymbol{z}_i, \boldsymbol{x}\big] + \text{const.} \tag{2.39}$$

This equation details how updating $q(\boldsymbol{z}_i)$ is a local operation, where a number of function-valued "messages" from the parents (the first term) and the children (the sum) are added together to form the update. To see under which conditions the messages can be computed reliably, we will have to dig into the subject of exponential families.

2.3 Exponential Families

The exponential family is a category of probability distributions that are particularly amenable to analysis and variational inference.

Definition 2.5 The exponential family consists of all distributions of the form,

$$p(\boldsymbol{x}) = h(\boldsymbol{x}) \exp\big(T(\boldsymbol{x})^\top \boldsymbol{\eta} - A(\boldsymbol{\eta})\big), \tag{2.40}$$

where $\boldsymbol{\eta}$ is the vector of natural parameters, $A(\cdot)$ is the log-normalizer, and $T(\cdot)$ is the vector-valued function of sufficient statistics of $\boldsymbol{x}$, while $h(\boldsymbol{x})$ is the base measure that the density reduces to if $\boldsymbol{\eta} = \boldsymbol{0}$.

This family of distributions might superficially appear to be of little practical interest, but it happens to include the majority of distributions in common use, including the Gaussian, the Bernoulli, the Poisson, the Gamma, the Beta, and many others.

The exponential family distributions can also be motivated in a constructive manner by considering a random quantity $\boldsymbol{x}$ which has known moments $\mathbb{E}[T(\boldsymbol{x})] = \boldsymbol{\mu}$. If all we know about the random quantity is the moment constraint, a reasonable model of $\boldsymbol{X}$ would be the maximum entropy distribution from the set of distributions with the fixed T-moment. This distribution can be shown to be an exponential family with sufficient statistics $T(\cdot)$, and there is a unique (minimal) exponential family with some $\boldsymbol{\eta}$ corresponding to every realizable choice of $\boldsymbol{\mu}$, which has led the latter quantity to be denoted the mean parameters as it provides an alternative parameterization of the exponential family class (Wainwright and Jordan 2008). To specify the natural parameters of this maximum entropy solution, we can start with the following relationship between the mean parameters $\boldsymbol{\mu}$ and natural parameters. $\boldsymbol{\eta}$

Lemma 2.6 *For a random variable following an exponential family distribution like the one defined in equation* (2.40)*, the mean and covariance of* $T(\boldsymbol{X})$ *can be found as,*

$$\mathbb{E}[T(\boldsymbol{X})] = \boldsymbol{\nabla}_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) \tag{2.41}$$

$$\mathrm{Cov}(T(\boldsymbol{X}), T(\boldsymbol{X})) = \boldsymbol{\nabla}^2_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) \tag{2.42}$$

Proof. As the log-normalizer, A can be defined explicitly as

$$A(\boldsymbol{\eta}) = \ln \int h(\boldsymbol{x}) \exp\big(T(\boldsymbol{x})^\top \boldsymbol{\eta}\big)\, \mathrm{d}\boldsymbol{x} \tag{2.43}$$

and if we take the derivative of this quantity with respect to $\boldsymbol{\eta}$, we get

$$\boldsymbol{\nabla} A(\boldsymbol{\eta}) = \frac{\int T(\boldsymbol{x}) h(\boldsymbol{x}) \exp\big(T(\boldsymbol{x})^\top \boldsymbol{\eta}\big)\, \mathrm{d}\boldsymbol{x}}{\int h(\boldsymbol{x}) \exp(T(\boldsymbol{x})^\top \boldsymbol{\eta})\, \mathrm{d}\boldsymbol{x}} = \mathbb{E}[T(\boldsymbol{X})] \tag{2.44}$$

Repeating the argument, we get

$$\boldsymbol{\nabla}^2 A(\boldsymbol{\eta}) = \boldsymbol{\nabla}\, \mathbb{E}[T(\boldsymbol{X})] = \mathbb{E}[(T(\boldsymbol{X}) - \nabla A(\boldsymbol{\eta}))T(\boldsymbol{X})] = \mathrm{Cov}(T(\boldsymbol{X}), T(\boldsymbol{X})) \tag{2.45}$$

proving the claim. □

While this result is extremely handy on its own, allowing us to turn expectation integrals into derivatives, it also lets us implicitly define the natural parameters

corresponding to a set of mean parameters as

$$\boldsymbol{\eta} = \left[\boldsymbol{\nabla}_{\boldsymbol{\eta}} A\right]^{-1}(\boldsymbol{\mu}). \tag{2.46}$$

There is a more amenable definition, though, in terms of the *Legendre transform* (or convex conjugate) $A^*(\boldsymbol{\mu})$ of $A(\boldsymbol{\eta})$, defined as

$$A^*(\boldsymbol{\mu}) = \sup_{\tilde{\boldsymbol{\eta}}} \boldsymbol{\mu}^\top \tilde{\boldsymbol{\eta}} - A(\tilde{\boldsymbol{\eta}}) = \boldsymbol{\mu}^\top \boldsymbol{\eta} - A(\boldsymbol{\eta}) \tag{2.47}$$

where the last equation follows as the optimization problem has critical points where $\boldsymbol{\mu} = \boldsymbol{\nabla}_{\boldsymbol{\eta}} A(\tilde{\boldsymbol{\eta}})$, which is solved by the uniquely matched pair $(\boldsymbol{\mu}, \boldsymbol{\eta})$ obeying lemma 2.6. The uniqueness is derived from lemma 2.6 which also happened to show that $A(\boldsymbol{\eta})$ is a convex function by virtue of the covariance being positive semi-definite. By simple pattern-matching, we note that we can also relate A^* to the entropy of the distribution as

$$\mathcal{H}_{\boldsymbol{x}} = -\,\mathbb{E}[\ln p(\boldsymbol{x})] = -\left(\boldsymbol{\eta}^\top \mathbb{E}[T(\boldsymbol{x})] - A(\boldsymbol{\eta}) + \mathbb{E}[\ln h(\boldsymbol{x})]\right) = -A^*(\boldsymbol{x}) + \mathcal{H}(h) \tag{2.48}$$

To retrieve the natural parameters, we can take the gradient of A^* with respect to $\boldsymbol{\mu}$, giving us the dual form of lemma 2.6 as

$$\boldsymbol{\eta} = \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}) \tag{2.49}$$

Given that the two parameterizations are interchangeable, we will finish this section by reparametrizing the exponential family of equation (2.40) using $\boldsymbol{\mu}$, yielding (Banerjee et al. 2005)

$$p(\boldsymbol{x}) = h(\boldsymbol{x}) \exp\left((\boldsymbol{\mu}^\top \boldsymbol{\eta} - A(\boldsymbol{\eta})) + (T(\boldsymbol{x}) - \boldsymbol{\mu})^\top \boldsymbol{\eta}\right) = \tag{2.50}$$

$$h(\boldsymbol{x}) \exp(A^*(T(\boldsymbol{x}))) \exp(-D_{A^*}(T(\boldsymbol{x}), \boldsymbol{\mu})) \tag{2.51}$$

where

$$D_F(\boldsymbol{x}, \boldsymbol{y}) = F(\boldsymbol{x}) - F(\boldsymbol{y}) - \boldsymbol{\nabla}_{\boldsymbol{y}} F(\boldsymbol{y})^\top (\boldsymbol{x} - \boldsymbol{y}) \tag{2.52}$$

is the general formula for a Bregman divergence associated with the convex function F.

2.3.1 Example: The Gaussian Distribution

The Gaussian remains the most important and widely used continuous distribution in use. The D-dimensional multivariate density function is

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{m})\right) \tag{2.53}$$

Expanding and reordering, we can get the natural parameterization

$$\frac{1}{(2\pi)^{\frac{D}{2}}}\exp\left(-\frac{1}{2}\boldsymbol{x}^\top\Sigma^{-1}\boldsymbol{x}+\boldsymbol{x}^\top\Sigma^{-1}\boldsymbol{m}-\frac{1}{2}\boldsymbol{m}^\top\Sigma^{-1}\boldsymbol{m}-\frac{1}{2}\ln|\boldsymbol{\Sigma}|\right). \tag{2.54}$$

Now, by using the trace properties, we can write the quadratic forms as inner products,

$$\boldsymbol{x}^\top\Sigma^{-1}\boldsymbol{x}=\operatorname{Tr}\left(\Sigma\boldsymbol{x}\boldsymbol{x}^\top\right)=\operatorname{vec}\left(\Sigma^{-1}\right)^\top\operatorname{vec}\left(\boldsymbol{x}\boldsymbol{x}^\top\right) \tag{2.55}$$

and then the Gaussian can be written in the standard exponential family form as

$$h(\boldsymbol{x})=\frac{1}{(2\pi)^{\frac{D}{2}}},\quad T(\boldsymbol{x})=\begin{pmatrix}\boldsymbol{x}\\ \operatorname{vec}\left(\boldsymbol{x}\boldsymbol{x}^\top\right)\end{pmatrix},\quad \boldsymbol{\eta}=\begin{pmatrix}\Sigma^{-1}\boldsymbol{m}\\ \operatorname{vec}\left(-\frac{1}{2}\Sigma^{-1}\right)\end{pmatrix}, \tag{2.56}$$

$$A=\frac{1}{2}\boldsymbol{m}^\top\Sigma^{-1}\boldsymbol{m}+\frac{1}{2}\ln|\Sigma|=-\frac{1}{4}\boldsymbol{\eta}_1^\top\operatorname{mat}(\boldsymbol{\eta}_2)^{-1}\boldsymbol{\eta}_1-\frac{1}{2}\ln\left|-2\operatorname{mat}(\boldsymbol{\eta}_2)\right|, \tag{2.57}$$

where we use $\operatorname{vec}(\cdot)$ and $\operatorname{mat}(\cdot)$ to denote vectorization and its inverse operation. From this we can see that Gaussians can be justified as maximum entropy distributions with known first and second moment. By taking the derivative of the log-normalizer using matrix calculus, we can find the

$$\boldsymbol{\mu}=\begin{pmatrix}-\frac{1}{2}\operatorname{mat}(\boldsymbol{\eta}_2)^{-1}\boldsymbol{\eta}_1\\ \operatorname{vec}\left(\frac{1}{4}\operatorname{mat}(\boldsymbol{\eta}_2)^{-1}\boldsymbol{\eta}_1\boldsymbol{\eta}_1^\top\operatorname{mat}(\boldsymbol{\eta}_2)^{-1}-\frac{1}{2}\boldsymbol{\eta}_2^{-1}\right)\end{pmatrix}=\begin{pmatrix}\boldsymbol{m}\\ \operatorname{vec}\left(\Sigma+\boldsymbol{m}\boldsymbol{m}^\top\right)\end{pmatrix} \tag{2.58}$$

which matches up with the expected results, i.e. the two first moments.

The conjugate log-normalizer in $\boldsymbol{m}$ and $\boldsymbol{\Sigma}$ standard parameters is

$$A^*=\boldsymbol{m}^\top\Sigma^{-1}\boldsymbol{m}-\frac{D}{2}-\frac{1}{2}\boldsymbol{m}^\top\Sigma^{-1}\boldsymbol{m}-\frac{1}{2}\boldsymbol{m}^\top\Sigma^{-1}\boldsymbol{m}-\frac{1}{2}\ln|\Sigma|=-\frac{1}{2}\ln|\Sigma|-\frac{D}{2} \tag{2.59}$$

and from that follows the entropy

$$\mathcal{H}(p)=-A^*-\mathbb{E}[\ln h(\boldsymbol{x})]=\frac{1}{2}\ln|\Sigma|+\frac{D}{2}+\frac{D}{2}\ln 2\pi=\frac{1}{2}\ln|2\pi e\Sigma| \tag{2.60}$$

Aside from the trivial integral over $h(\boldsymbol{x})$, note that all of the above results have been derived without resorting to integrals.

2.3.2 Conjugate Priors

While exponential families are eminently tractable as demonstrated above, the real challenge of Bayesian inference is integrating information from the likelihood

with the prior. Recall that Bayes' theorem says that the posterior is proportional to the product of the prior and the likelihood,

$$p(\boldsymbol{\eta}|\{\boldsymbol{x}_n\}_{n=1}^N) \propto \prod_{n=1}^N p(\boldsymbol{x}_n|\boldsymbol{\eta})p(\boldsymbol{\eta}). \tag{2.61}$$

So if both the likelihood and the prior are distributed according to exponential family distributions, does this imply that the posterior is also distributed according to an exponential family? Taking the logarithm of the right-hand side and assuming that the prior has base measure h_0, sufficient statistics $T_0(\cdot)$, and natural parameters $\boldsymbol{\gamma}$,

$$\sum_{n=1}^N \ln p(\boldsymbol{x}_n|\boldsymbol{\eta}) + \ln p(\boldsymbol{\eta}) = \begin{pmatrix} \boldsymbol{\eta} \\ -A(\boldsymbol{\eta}) \end{pmatrix}^\top \begin{pmatrix} \sum_{n=1}^N T(\boldsymbol{x}_n) \\ N \end{pmatrix} + \boldsymbol{\gamma}^\top T_0(\boldsymbol{\eta}) + \ln h_0(\boldsymbol{\eta}) + \text{const.} \tag{2.62}$$

So an exponential family prior with sufficient statistics T_0 induces an exponential family posterior with sufficient statistics $(T_0(\boldsymbol{\eta}), \boldsymbol{\eta}, -A(\boldsymbol{\eta}))$. As such, combinations of arbitrary likelihoods and arbitrary priors can sweep out exponential families with all kinds of ungainly sufficient statistics. The conjugate prior to a likelihood is any prior such that the posterior has the same sufficient statistics as the prior. The canonical conjugate prior is the one where

$$\ln p(\boldsymbol{\eta}) = \begin{pmatrix} \boldsymbol{\eta} \\ -A(\boldsymbol{\eta}) \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\gamma} \\ \nu \end{pmatrix} - A_0(\boldsymbol{\gamma}, \nu) + \ln h_0(\boldsymbol{\eta}), \tag{2.63}$$

inducing a posterior of the form,

$$\ln p(\boldsymbol{\eta}|\{\boldsymbol{x}_n\}_{n=1}^N) = \begin{pmatrix} \boldsymbol{\eta} \\ -A(\boldsymbol{\eta}) \end{pmatrix}^\top \begin{pmatrix} \sum_{n=1}^N T(\boldsymbol{x}_n) + \boldsymbol{\gamma} \\ N + \nu \end{pmatrix} - A_0(\boldsymbol{\gamma}, \nu) + \ln h_0(\boldsymbol{\eta}), \tag{2.64}$$

where we have dedicated a special natural parameter ν to the sufficient statistic $-A(\boldsymbol{\eta})$. A classical interpretation of the conjugate prior is that it simulates a series of pseudo-observations: ν is added to the datapoint counter N, and $\boldsymbol{\gamma}$ is added to the accumulated sufficient statistics, so that the combined effect is equivalent to having observed ν new datapoints with average sufficient statistic $\frac{1}{\nu}\boldsymbol{\gamma}$.

2.3.3 Conjugate Models and Variational Message Passing

In section 2.2.3 we detailed a scheme for inference when the q model factorized, but ended at an impassé as we were left with a number of tough expectations in

the update formula,

$$\ln q^*(\boldsymbol{z}_i) = \mathbb{E}[\ln p(\boldsymbol{z}_i|\mathrm{pa}_i)|\boldsymbol{z}_i, \boldsymbol{x}] + \sum_{j\in \mathrm{ch}_i} \mathbb{E}\big[\ln p(\boldsymbol{z}_j|\mathrm{pa}_j)|\boldsymbol{z}_i, \boldsymbol{x}\big] + \text{const.} \tag{2.65}$$

If we assume that each conditional distribution takes the shape of an exponential family distribution with $\boldsymbol{\eta}_i$ being a function of the conditioning set $\boldsymbol{\eta}_i(\mathrm{pa}_i)$,

$$\mathbb{E}\big[\ln p(\boldsymbol{z}_j|\mathrm{pa}_j)\big] = \mathbb{E}[\ln h(\boldsymbol{z}_j)] + \begin{pmatrix}\mathbb{E}[T(\boldsymbol{z}_j)]\\ 1\end{pmatrix}^\top \begin{pmatrix}\mathbb{E}[\boldsymbol{\eta}_j]\\ -\mathbb{E}[A_j(\boldsymbol{\eta}_j)]\end{pmatrix}, \tag{2.66}$$

where we used the independence assumption to split the expectation across terms, then the update formula for $\ln q^*(\boldsymbol{z}_i)$ can be stated as

$$h(\boldsymbol{z}_i) + T(\boldsymbol{z}_i)^\top \mathbb{E}[\boldsymbol{\eta}_i|\boldsymbol{x}] + \sum_{j\in \mathrm{ch}_i} \begin{pmatrix}\boldsymbol{\mu}_j\\ 1\end{pmatrix}^\top \begin{pmatrix}\mathbb{E}[\boldsymbol{\eta}_j|\boldsymbol{z}_i, \boldsymbol{x}]\\ -\mathbb{E}[A_j(\boldsymbol{\eta}_j)|\boldsymbol{z}_i, \boldsymbol{x}]\end{pmatrix}. \tag{2.67}$$

So we can ask ourselves what assumptions we have to make to ensure that this is tractable.

Recall that the natural parameter was a function of the conditioning set $\boldsymbol{\eta}_i(\mathrm{pa}_i)$. The simplest scenario is the one where this function is the identity, and each variable only has one parent, so that the graphical model is tree-structured and $\boldsymbol{\eta}_i = \boldsymbol{z}_{\mathrm{pa}_i}$. The update formula then becomes

$$h(\boldsymbol{z}_i) + T(\boldsymbol{z}_i)^\top \mathbb{E}\big[\boldsymbol{z}_{\mathrm{pa}_i}|\boldsymbol{x}\big] + \sum_{j\in \mathrm{ch}_i} \begin{pmatrix}\boldsymbol{\mu}_j\\ 1\end{pmatrix}^\top \begin{pmatrix}\boldsymbol{z}_i\\ -A_i(\boldsymbol{z}_i)\end{pmatrix} \tag{2.68}$$

If $T(\boldsymbol{z}_i) = \begin{pmatrix}\boldsymbol{z}_i\\ -A_i(\boldsymbol{z}_i)\end{pmatrix}$, then the updated distribution $q(\boldsymbol{z}_i)$ has the same functional form as its prior. This is the same set of sufficient statistics we found for the conjugate prior, so for tree-structured graphical models we require that the parent of each node is a conjugate prior to it.

For more general graphs where variables can have multiple parent nodes, we will have to consider something called variational message passing (Winn and Bishop 2005). If the terms coming from the sum in equation (2.67) happened to be consistent with the sufficient statistics of $\boldsymbol{z}_i$, in the sense that

$$\begin{pmatrix}\boldsymbol{\eta}_j\\ -A_j(\boldsymbol{\eta}_j)\end{pmatrix} = \boldsymbol{M}_j(\mathrm{pa}_j/\{\boldsymbol{z}_i\})T(\boldsymbol{z}_i) \tag{2.69}$$

for some matrix-valued function of the co-parents $\boldsymbol{M}_j(\cdot)$, then the message passing formula would add together to

$$h(\boldsymbol{z}_i) + T(\boldsymbol{z}_i)^\top \mathbb{E}\left[\boldsymbol{\eta}_i + \sum_{j \in \mathrm{ch}_i} \boldsymbol{M}_j(\mathrm{pa}_j/\{\boldsymbol{z}_i\})^\top \begin{pmatrix} \boldsymbol{\mu}_j \\ 1 \end{pmatrix}\right]. \tag{2.70}$$

which is a member of the same exponential family as the prior on $\boldsymbol{z}_i$ with sufficient statistics $T(\boldsymbol{z}_i)$. Note that the same decomposition should be possible for all other parents of j as well. Notably, this means that equation (2.69) should hold, with the matrix changing with index k:

$$\begin{pmatrix} \boldsymbol{\eta}_j \\ -A_j(\boldsymbol{\eta}_j) \end{pmatrix} = \boldsymbol{M}_{j,k}(\mathrm{pa}_j/\{\boldsymbol{z}_k\})T(\boldsymbol{z}_k), \quad \forall k \in \mathrm{pa}_j \tag{2.71}$$

This means that $\boldsymbol{\eta}_j$ must be a multi-linear function of the sufficient statistics of all the P parents

$$\begin{pmatrix} \boldsymbol{\eta}_j \\ -A_j(\boldsymbol{\eta}_j) \end{pmatrix} = M_j(T_{[\mathrm{pa}_j]_1}(\boldsymbol{z}_{[\mathrm{pa}_j]_1}), \ldots, T_{[\mathrm{pa}_j]_P}(\boldsymbol{z}_{[\mathrm{pa}_j]_P})) \tag{2.72}$$

and since expectations are linear operators, we can easily compute expectations over M as we know the expected sufficient statistics $\boldsymbol{\mu}$ for each of its inputs,

$$\mathbb{E}\Big[\boldsymbol{\eta}_j \Big| T_{[\mathrm{pa}_j]_i}(\boldsymbol{z}_{[\mathrm{pa}_j]_i})\Big] = M_j(\boldsymbol{\mu}_{[\mathrm{pa}_j]_1}, \ldots, T_{[\mathrm{pa}_j]_i}(\boldsymbol{z}_{[\mathrm{pa}_j]_i}), \ldots, \boldsymbol{\mu}_{[\mathrm{pa}_j]_P}) \tag{2.73}$$

Together, this allows us to compute $q^*(\boldsymbol{z}_i)$ from equation (2.67) by aggregating info from the mean parameters $\boldsymbol{\mu}_j$ of the Markov blanket $j \in \mathrm{mb}_i$ using the multi-linear aggregating functions M_i.

2.3.4 Stochastic Variational Inference

Conjugate exponential family models do not just make the message passing updates easier, they can also make gradient computations much simpler. If we focus on a single factor $q(\boldsymbol{z}_k; \boldsymbol{\eta}_k)$ assumed to be in the exponential family, then the ELBO gradient from equation (2.11) becomes

$$\nabla_{\boldsymbol{\eta}_k} \mathcal{L}(\boldsymbol{\eta}) = \mathbb{E}\left[(\nabla_{\boldsymbol{\eta}_k} \ln q(\boldsymbol{z})) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z}; \boldsymbol{\eta})}\right] = \tag{2.74}$$

$$\mathbb{E}_{\boldsymbol{z}_k}\left[(T(\boldsymbol{z}_k)) - \mathbb{E}[T(\boldsymbol{z}_k)]\right)\left(\mathbb{E}[\ln p(\boldsymbol{z}, \boldsymbol{x})|\boldsymbol{z}_k] - T(\boldsymbol{z}_k)^\top \boldsymbol{\eta}_k - \ln h(\boldsymbol{z}_k))\right)\Big], \tag{2.75}$$

where we could discard all terms (i.e. the log-normalizer and other factors of q) that were constant with respect to $\boldsymbol{\eta}_k$ and were multiplied by the factor $T(\boldsymbol{z}_k) - \mathbb{E}[T(\boldsymbol{z}_k)]$ as it has expectation 0. Now, we can recognize the expected

$\ln p$ term as the optimal local update to $q(\boldsymbol{z}_k)$ from equation (2.36). If p is a tree-shaped conjugate exponential family model, then we know that $q^*(\boldsymbol{z}_k)$ is of the same functional form as the conditional prior $p(\boldsymbol{z}_k|\mathrm{pa}_k)$ as corroborated by equation (2.68),

$$\ln q^*(\boldsymbol{z}) + \text{const.} = \mathbb{E}[\ln p(\boldsymbol{x}, \boldsymbol{z})|\boldsymbol{z}_k] = \ln h(\boldsymbol{z}_k) + T(\boldsymbol{z}_k)^\top \boldsymbol{\eta}_k^* - NA(\boldsymbol{\eta}_k). \quad (2.76)$$

Plugging this result back into equation (2.74) we get the gradient formula

$$\mathbb{E}_{\boldsymbol{z}_k}[(T(\boldsymbol{z}_k)) - \mathbb{E}[T(\boldsymbol{z}_k)])\, T(\boldsymbol{z}_k)]\, (\boldsymbol{\eta}_k^* - \boldsymbol{\eta}_k), \quad (2.77)$$

which we see is easily set to 0 by updating $q(\boldsymbol{z}_k)$ to be $q^*(\boldsymbol{z}_k)$, corroborating that it is the optimal update. This equation still requires computing an expectation, but we actually already computed the value of this expectation in lemma 2.6, where we saw that the covariance of the sufficient statistics is simply the Hessian of the log-normalizer,

$$\mathrm{Cov}(T(\boldsymbol{z}_k), T(\boldsymbol{z}_k)) = \boldsymbol{\nabla}^2_{\boldsymbol{\eta}_k} A(\boldsymbol{\eta}_k). \quad (2.78)$$

For exponential families, the covariance of the sufficient statistics happen to be equal to the Fisher information matrix $\mathcal{I}$, so we can write the full gradient compactly as,

$$\boldsymbol{\nabla}_{\boldsymbol{\eta}_k} \mathcal{L}(\boldsymbol{\eta}) = \mathcal{I}_{q_k}(\boldsymbol{\eta}_k^* - \boldsymbol{\eta}_k). \quad (2.79)$$

2.3.4.1 Natural Gradients

Computing a Hessian can be an inconvenience, but as argued by Hoffman, Blei, et al. (2013) we can actually get around that by using natural gradients. Ordinary gradients try to find the direction of steepest ascent using the standard Euclidean norm to measure step distance; the natural gradient is what we get if we use a distance measure more appropriate for the parameter space. If we use symmetrized Kullback-Leibler divergence, the natural gradient $\tilde{\boldsymbol{\nabla}}\mathcal{L}$ is a preconditioned Euclidean gradient,

$$\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\eta}_k} \mathcal{L}(\boldsymbol{\eta}) = \mathcal{I}_{q_k}^{-1} \boldsymbol{\nabla}_{\boldsymbol{\eta}_k} \mathcal{L}(\boldsymbol{\eta}) \quad (2.80)$$

where $\mathcal{I}$ is again the Fisher information matrix (Hoffman, Blei, et al. 2013). Very conveniently, this preconditioner cancels exactly with the Fisher matrix pre-factor of the original gradient, so the natural gradient can be stated succinctly as (Hoffman, Blei, et al. 2013),

$$\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\eta}_k} \mathcal{L} = \boldsymbol{\eta}_k^* - \boldsymbol{\eta}_k. \quad (2.81)$$

This update shows that a natural gradient step of length 1 matches exactly with an optimal local update (Sato 2001; Hoffman, Blei, et al. 2013). To complete the

description of the update, we can find a formula for $\boldsymbol{\eta}^*$ by returning to section 2.2.3. Of special interest is again the tree-shaped conjugate exponential family graphical models, where by extracting the relevant terms from equation (2.68),

$$\boldsymbol{\eta}_i^* = \mathbb{E}\big[\boldsymbol{z}_{\mathrm{pa}_i}|\boldsymbol{x}\big] + \sum_{j\in\mathrm{ch}_i}\begin{pmatrix}\boldsymbol{\mu}_j\\1\end{pmatrix} \tag{2.82}$$

2.3.4.2 Stochastic Updates

Note that equation (2.82) (and the natural gradient of equation (2.81) by extension) decomposes nicely as a sum over the child indices j. A benefit of sum decompositions is that we can also construct stochastic estimators of the gradient,

$$\hat{\boldsymbol{\eta}}_i^* = \mathbb{E}\big[\boldsymbol{z}_{\mathrm{pa}_i}|\boldsymbol{x}\big] + N\begin{pmatrix}\boldsymbol{\mu}_{\mathcal{I}}\\1\end{pmatrix} \tag{2.83}$$

where $\mathcal{I}$ is a random index, sampled uniformly from the indices of the children ch_i. As this estimator is demonstrably unbiased, we can use it in any stochastic gradient algorithm, trading off a larger number of iterations to convergence for a lower per-iteration cost (Robbins and Monro 1951; Bottou 2010). Another advantage of only having to evaluate one child node is that we can get away with only lazily updating our parameters, which is the real key to the algorithmic procedure known as Stochastic Variational Inference (Hoffman, Blei, et al. 2013). If we want to update node i optimally, we should first update all of its children conditioned on i. But if there is a large number of children, this update will carry a high computational cost. If we use the stochastic update, we can simply pretend that we updated all of the child nodes — we only actually have to compute the single child node corresponding to $\mathcal{I}$, as the rest do not enter into the stochastic update. So we can perform (stochastic) optimal updates, with the iteration cost scaling in the number of samples used for the stochastic estimator ($\mathcal{O}(1)$) instead of in the number of children ($\mathcal{O}(N)$).

2.4 Lower Bound Algebra

Having covered the basics of variational inference in conjugate exponential and black-box models, we can consider how to improve on the basic design by making inference more efficient or by expanding to more complicated models or approximations.

2.4.1 Variational Approximations with Auxiliary Random Variables

Outside of the black-box paradigm, we are restricted to using q models that are tractable in the sense that we can compute both their expectation over the log-likelihood and their entropy. A number of authors have demonstrated how the complexity of the variational model can be increased by introducing auxiliary variables $\boldsymbol{u}$ into the approximation (Agakov and Barber 2004; Salimans and Knowles 2013; Ranganath, Tran, and Blei 2015). The canonical example of an auxiliary variable variational approximation is

$$q(\boldsymbol{z}) = \int q(\boldsymbol{z}|\boldsymbol{u})q(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u} \tag{2.84}$$

To get a bound in both $\boldsymbol{z}$ and $\boldsymbol{u}$, the following theorem can be used (Salimans and Knowles 2013; Ranganath, Tran, and Blei 2015).

Theorem 2.7 *For a variational approximation $q(\boldsymbol{z}) = \int q(\boldsymbol{z}|\boldsymbol{u})q(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u}$, the following bound is equivalent to the ELBO for $q(\boldsymbol{z})$*

$$\mathcal{L}(q) = \mathbb{E}_{q(\boldsymbol{u},\boldsymbol{z})}[\ln \tilde{p}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{u}) - \ln q(\boldsymbol{z},\boldsymbol{u})] \tag{2.85}$$

where $\tilde{p}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{u}) = p(\boldsymbol{x},\boldsymbol{z})q(\boldsymbol{u}|\boldsymbol{z})$, with $q(\boldsymbol{u}|\boldsymbol{z}) = q(\boldsymbol{u},\boldsymbol{z})/q(\boldsymbol{z})$

Proof. Simply decompose $\ln q(\boldsymbol{z},\boldsymbol{u}) = \ln q(\boldsymbol{z}) + \ln q(\boldsymbol{u}|\boldsymbol{z})$ to cancel out the extra factor in the modified joint $\tilde{p}$. □

If the auxiliary conditional $q(\boldsymbol{u}|\boldsymbol{z})$ is intractable, it can be further approximated with its own independent variational approximation $r(\boldsymbol{u}|\boldsymbol{z})$ (Ranganath, Tran, and Blei 2015).

Lemma 2.8 *For any two distributions $q(\boldsymbol{u},\boldsymbol{z})$ and $r(\boldsymbol{u}|\boldsymbol{z})$*

$$\mathbb{E}_{q(\boldsymbol{u},\boldsymbol{z})}[\ln q(\boldsymbol{u}|\boldsymbol{z})] \geq \mathbb{E}_{q(\boldsymbol{u},\boldsymbol{z})}[\ln r(\boldsymbol{u}|\boldsymbol{z})] \tag{2.86}$$

Proof. Add $\mathbb{E}[\ln r(\boldsymbol{u}|\boldsymbol{z}) - \ln r(\boldsymbol{u}|\boldsymbol{z})]$ to the left-hand side to get

$$\mathbb{E}[\ln r(\boldsymbol{u}|\boldsymbol{z})] + \mathbb{E}\left[\ln \frac{q(\boldsymbol{u}|\boldsymbol{z})}{r(\boldsymbol{u}|\boldsymbol{z})}\right] = \mathbb{E}[\ln r(\boldsymbol{u}|\boldsymbol{z})] + \mathrm{KL}\left(q(\boldsymbol{u},\boldsymbol{z})\|\tilde{r}(\boldsymbol{u},\boldsymbol{z})\right) \tag{2.87}$$

where $\tilde{r}(\boldsymbol{u},\boldsymbol{z}) = r(\boldsymbol{u}|\boldsymbol{z})q(\boldsymbol{z})$. Since the divergence is non-negative, the bound follows. □

Applying the above inequality to $\mathcal{L}$ then yields a new bound for hierarchical approximations (Ranganath, Tran, and Blei 2015).

Theorem 2.9 *Define a bound*

$$\mathcal{L}_H(q,r) = \mathbb{E}[\ln p(\boldsymbol{x},\boldsymbol{z})] - \mathbb{E}[\ln q(\boldsymbol{u}) + \ln q(\boldsymbol{z}|\boldsymbol{u}) - \ln r(\boldsymbol{u}|\boldsymbol{z})] \tag{2.88}$$

then $\mathcal{L}(q) \geq \mathcal{L}_H(q,r)$ *with the gap being* $\mathrm{KL}\left(q(\boldsymbol{u},\boldsymbol{z}) \| \tilde{r}(\boldsymbol{u},\boldsymbol{z})\right)$.

Proof. Apply lemma 2.8 to the expanded bound in equation (2.85). The gap follows from the proof of the lemma. □

2.4.2 Collapsed Bounds for Optimization Space Reduction

Probabilistic models can involve a number of latent nuisance parameters $\boldsymbol{u}$ which are necessary to formulate the model, but irrelevant for the final analysis. Even if this is not the case, we can sometimes get away with estimating the posterior on a subset of the latent variables, and then use the approximation to formulate a posterior on the remaining variables analytically. If our model takes the factorized form

$$p(\boldsymbol{x},\boldsymbol{z},\boldsymbol{u}) = p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})p(\boldsymbol{u}), \tag{2.89}$$

the ideal solution would be to simply use

$$p(\boldsymbol{x},\boldsymbol{z}) = \int p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})p(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u}\,, \tag{2.90}$$

for calculating the posterior and any other inference tasks. The marginalization integral can be tractable, e.g. for conjugate exponential family models, but marginalized exponential families are rarely exponential families themselves (Hensman, Rattray, and Lawrence 2012). As a consequence, it is difficult to perform variational inference on marginalized models unless we use black-box methods.

The KL-corrected bound is an alternative bound which is only over the latent variables $\boldsymbol{z}$. It does not marginalize out the nuisance parameters, but rather assumes that $q(\boldsymbol{z},\boldsymbol{u}) = q(\boldsymbol{z})q(\boldsymbol{u})$ and then replaces $q(\boldsymbol{u})$ with its optimal setting $q^*(\boldsymbol{u})$ from equation (2.36). The KL-corrected bound thus achieves a reduction in the number of parameters to be optimized for.

An enlightening derivation of the KL-corrected bound follows from first calculating the auxiliary bound

$$\ln p(\boldsymbol{x}|\boldsymbol{u}) \geq \mathcal{L}_{\boldsymbol{u}}(q) = \mathbb{E}_{q(\boldsymbol{z})}[\ln p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})] - \mathbb{E}_{q(\boldsymbol{z})}[\ln q(\boldsymbol{z})]\,, \tag{2.91}$$

which is the standard ELBO for the conditional model $p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})$. The lower bound property is preserved under monotonic transformations, so we use the transform $\ln \mathbb{E}_{p(\boldsymbol{u})}[\exp(\cdot)]$ on both sides, yielding the full KL-corrected bound,

$$\ln p(\boldsymbol{x}) \geq \ln \int p(\boldsymbol{u}) \exp(\mathcal{L}_{\boldsymbol{u}}(q))\, \mathrm{d}\boldsymbol{u} = \tag{2.92}$$

$$\ln \int p(\boldsymbol{u}) \exp\big(\mathbb{E}_{q(\boldsymbol{z})}[\ln p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})]\big)\, \mathrm{d}\boldsymbol{u} - \mathbb{E}[\ln q(\boldsymbol{z})]. \tag{2.93}$$

Two results make this setup elegant. First, we know that $q^*(\boldsymbol{u}) \propto \exp\big(\mathbb{E}_{q(\boldsymbol{z})}[\ln p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u})]\big)$ from earlier. Knowing the two bounds, we can instead write

$$q^*(\boldsymbol{u}) = p(\boldsymbol{u}) \exp(\mathcal{L}_{\boldsymbol{u}}(q(\boldsymbol{z})) - \mathcal{L}_{KL}(q(\boldsymbol{z}))). \tag{2.94}$$

The auxiliary bound $\mathcal{L}_{\boldsymbol{u}}$ modulates the prior through its dependence on $\boldsymbol{u}$, while $\mathcal{L}_{KL}$ is the normalization constant. This means that we can retrieve the optimal normalized density function for $\boldsymbol{u}$ at any time using the value of the bounds.

Second, whereas other alternative bounds fail to characterize what divergence they are minimizing we can state the following theorem:

Theorem 2.10 *The KL-corrected bound of $p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u})$ is related to the mean-field ELBO $\mathcal{L}_{\mathrm{MF}}$ factorizing over $\boldsymbol{u}$ and $\boldsymbol{z}$ by*

$$\mathcal{L}_{KL} = \mathcal{L}_{\mathrm{MF}} + \mathrm{KL}\left(q(\boldsymbol{z}) \| q^*(\boldsymbol{z})\right) \tag{2.95}$$

Proof. Using equation (2.94), we can expand the KL divergence as

$$\mathrm{KL}\left(q(\boldsymbol{z}) \| q^*(\boldsymbol{z})\right) = \mathbb{E}_{q(\boldsymbol{u})}\left[\ln \frac{q(\boldsymbol{u})}{p(\boldsymbol{u})} - \mathcal{L}_{\boldsymbol{u}}(q)\right] + \mathcal{L}_{KL}(q) \tag{2.96}$$

Plugging in the definition for $\mathcal{L}_{\boldsymbol{u}}(q)$ from equation (2.91), we can simplify the expression,

$$\mathbb{E}_{q(\boldsymbol{u})}\left[\ln \frac{q(\boldsymbol{u})q(\boldsymbol{z})}{p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u})}\right] + \mathcal{L}_{KL}(q) = -\mathcal{L}_{\mathrm{MF}}(q(\boldsymbol{z}), q(\boldsymbol{u})) + \mathcal{L}_{KL}(q(\boldsymbol{z})) \tag{2.97}$$

Adding the mean-field bound then proves the theorem. □

The KL-corrected bound is then an upper bound to the mean-field ELBO, which saturates when $q(\boldsymbol{u}) = q^*(\boldsymbol{u})$.

2.5 Biased Bounds for Model Selection

While we have mostly used variational bounds to find the optimal choice of approximation q, it is also frequently used for model selection. Recall that model selection is the process of finding the set of hyperparameters $\boldsymbol{\alpha}$ so that the actual probabilistic model $p(\boldsymbol{x};\boldsymbol{\alpha})$ generalizes as well as possible. Often, this is accomplished by maximizing the log-evidence $\ln p(\boldsymbol{x};\boldsymbol{\alpha})$ for fixed data $\boldsymbol{x}$, in a procedure known as empirical Bayes (Bernardo and Smith 2000; Bishop 2006). Unfortunately, the evidence is exactly the intractable normalization constant we are using approximate inference to avoid, making it hard to use it in practice.

As the ELBO is a lower bound of the log-evidence, it is tempting to use it as a proxy as maximizing

$$\mathcal{L}(q;\boldsymbol{\alpha}) \leq \ln p(\boldsymbol{x};\boldsymbol{\alpha}), \tag{2.98}$$

with respect to the hyperparameters $\boldsymbol{\alpha}$ should push the log-evidence upwards. This is a strong assumption though, as the real relationship was given in lemma 2.2 as

$$\mathcal{L}(q;\boldsymbol{\alpha}) = \ln p(\boldsymbol{x};\boldsymbol{\alpha}) - \mathrm{KL}\left(q(\boldsymbol{z}) \| p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\alpha})\right). \tag{2.99}$$

Maximizing the lower bound could push the log-evidence up, but the divergence term also plays in. It is possible for the log-evidence to go down, while the KL divergence increases proportionally to close the gap, yielding a net positive gain. A realistic scenario could be that we pick a set of hyperparameters diffusing and smoothening the model density, worsening the model fit, but making it easier to approximate. A number of authors have thus pursued lower bounds with a tighter fit to the model evidence.

The central tool in this construction is Jensen's inequality, which states that for concave functions like $f(\cdot) = \ln(\cdot)$,

$$\mathbb{E}[f(g(x))] \leq f(\mathbb{E}[g(x)]). \tag{Jensen's Inequality}$$

Plugging in the logarithm for f and the probability ratio p/q for g, we get

$$\mathbb{E}_q\left[\ln\left(\frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}\right)\right] \leq \ln \mathbb{E}_q\left[\frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}\right] = \ln \int p(\boldsymbol{x},\boldsymbol{z})\,\mathrm{d}\boldsymbol{z} = \ln p(\boldsymbol{x}), \tag{2.100}$$

which is another proof of the lower bounding property. The key feature used in the argument is that $\frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}$ is an unbiased estimator of the log-evidence $\ln p(\boldsymbol{x})$ when $\boldsymbol{z} \sim q$. Extrapolating from this, we could replace the density ratio with any other parametric unbiased estimator $\mathbb{E}_{q(\boldsymbol{z})}[\hat{p}(\boldsymbol{x},\boldsymbol{z},q;\boldsymbol{\alpha})] = p(\boldsymbol{x};\boldsymbol{\alpha})$, generating a family of Monte Carlo Objectives (Maddison et al. 2017)

$$\mathcal{L}_{\mathrm{MCO}} = \mathbb{E}[\ln \hat{p}(\boldsymbol{x},\boldsymbol{z},q;\boldsymbol{\alpha})]. \tag{2.101}$$

One important member of this group of bounds is the Importance Weighted ELBO (IW-ELBO) which precedes the more general definition above (Burda, Grosse, and Salakhutdinov 2015). More commonly known as the IWAE (importance-weighted auto-encoder) bound due to it initially being used to train auto-encoders, it is found by simply taking multiple samples $\boldsymbol{z}^{(i)} \sim q$ and expanding the implicit unbiased estimator of the standard ELBO in a natural fashion as

$$\mathcal{L}_S(q) = \mathbb{E}\left[\ln \frac{1}{S}\sum_{i=1}^{S} \frac{p(\boldsymbol{x}, \boldsymbol{z}^{(i)})}{q(\boldsymbol{z}^{(i)})}\right] \tag{2.102}$$

This reduces to the ELBO for $S = 1$, and the authors go on to show that (Burda, Grosse, and Salakhutdinov 2015, Theorem 1),

$$\ln p(\boldsymbol{x}) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k, \tag{2.103}$$

demonstrating a natural progression of the bounds. We note that the bound grows tighter with decreasing variance, a notion that was later formalized for general Monte Carlo Objectives (Maddison et al. 2017).

Going back to Jensen's inequality, we could also change the choice of concave function. Defining

$$V(\boldsymbol{z}; \boldsymbol{\alpha}) = \ln q(\boldsymbol{z}) - \ln p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\alpha},) \tag{2.104}$$

we can write the evidence as,

$$p(\boldsymbol{x}) = \mathbb{E}[\exp(-V(\boldsymbol{z}; \boldsymbol{\alpha}))]. \tag{2.105}$$

For a concave function f with $f(x) \leq x$ when $x > 0$, we can then define the f-ELBO as (Bamler et al. 2017),

$$\mathbb{E}[f(\exp(-V(\boldsymbol{z}; \boldsymbol{\alpha})))] \leq f(p(\boldsymbol{x})) \leq p(\boldsymbol{x}). \tag{2.106}$$

For f being the identity, we get zero bias as it reduces to importance sampling of $\mathbb{E}_{p(\boldsymbol{z})}[p(\boldsymbol{x}|\boldsymbol{z})]$ using q, but importance sampling often suffers from large variance. If f is the logarithm, on the other hand, then we get the ELBO which has lower variance than importance sampling due to the canceling out of the exponential function, but suffers from bias. A proposed bound lying between the two extremes is the perturbative variational bound employing the function implicitly defined by

$$f_{V_0}(e^{-V}) = e^{-V_0}\left(1 + (V_0 - V) + \frac{1}{2}(V_0 - V)^2 + \frac{1}{6}(V_0 - V)^3\right), \tag{2.107}$$

which effectively swaps the exponential function for a third order Taylor approximation, reducing the growth rate and the bias simultaneously (Bamler et al. 2017).

While the biased bounds are appropriate proxies for optimizing $\boldsymbol{\alpha}$, it is not clear whether we can use them to find q as well, like how the original ELBO is a surrogate for the KL divergence via equation (2.99). As such, it might not be appropriate to maximize the new bound with respect to q; we are quite sure not to be optimizing the KL divergence anymore, and if we are still minimizing something divergence-like, it is unclear what properties it preserves. In a recent paper, it was shown that every Monte Carlo objective actually does minimize the KL divergence, just not with respect to the original posterior. They write (Le et al. 2018, Claim 1),

Lemma 2.11 *Given a non-negative unbiased estimator $\hat{p}(\boldsymbol{x}, \boldsymbol{z}, q; \boldsymbol{\alpha}) \geq 0$ of the normalizing constant $p(\boldsymbol{x}; \boldsymbol{\alpha})$ where $\boldsymbol{z}$ is distributed according to the proposal distribution $q(\boldsymbol{z})$, the following holds:*

$$\mathbb{E}_q[\ln \hat{p}(\boldsymbol{x}, \boldsymbol{z}, q; \boldsymbol{\alpha})] = \ln p(\boldsymbol{x}; \boldsymbol{\alpha}) - \mathrm{KL}\left(q \| p^*\right), \tag{2.108}$$

where

$$p^*(\boldsymbol{z}) = \frac{q(\boldsymbol{z})\hat{p}(\boldsymbol{x}, \boldsymbol{z}, q; \boldsymbol{\alpha})}{p(\boldsymbol{x}; \boldsymbol{\alpha})} \tag{2.109}$$

is the implied normalized posterior.

For the standard ELBO where $\hat{p}(\boldsymbol{x}, \boldsymbol{z}, q; \boldsymbol{\alpha}) = \frac{p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\alpha})}{q(\boldsymbol{z})}$ the proxy posterior p^* is actually the true posterior, consistent with our previous beliefs (Le et al. 2018). For the IW-ELBO, we can take $q(\{\boldsymbol{z}_K\}_{k=1}^S) = \prod_{k=1}^K q(\boldsymbol{z}^K)$ and get

$$\mathrm{KL}\left(\prod_{k=1}^K q(\boldsymbol{z}^{(k)}) \middle\| \frac{1}{K}\sum_{k=1}^K \frac{\prod_{\ell=1}^K q(\boldsymbol{z}^{(\ell)})}{q(\boldsymbol{z}^{(k)})} p(\boldsymbol{z}^{(k)}|\boldsymbol{x})\right). \tag{2.110}$$

This divergence is still minimized at $q = p$, but as K grows each marginal $\boldsymbol{z}^{(k)}$ of the proxy posterior becomes a mix of $K - 1$ copies of q and a single copy of the true posterior $p(\boldsymbol{z}^{(k)}|\boldsymbol{x})$. In the limit, the true posterior will have a vanishingly small influence on the shape of the proxy posterior. This illustrates that the IW bound is not really an appropriate objective function, despite its other properties (Rainforth et al. 2018). As a curiosity, we note that the proxy posterior above is reminiscent of the proxy posterior used in pseudo-extended MCMC, except with q used instead of smoothened posteriors (Nemeth et al. 2017).

2.6 Contribution

Our own contributions to the field of variational inference are two-fold, split across our two papers *Difference-of-convex optimization for variational KL-*

corrected inference in Dirichlet process mixtures and *Matrix Product states for inference in discrete probabilistic models*. The first is strongly rooted in the traditional approach to variational inference, where we explore the properties of the KL-corrected bound in the special case where the bound has a so-called difference-of-convex structure. Just like the optimal coordinate-ascent steps turned out to be interpretable in terms of natural gradients (Sato 2001), we show that applying the convex-concave procedure by Yuille (2002) to the KL-corrected bound leads to a novel fixed-point algorithm. For the product of N independent categoricals on K categories that we used, $q(\boldsymbol{Z}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mu_{nk}^{z_{nk}}$, with parameter $\boldsymbol{\mu}$, the fixed point formula turned out as,

$$\boldsymbol{\mu}_{t+1} = \text{softmax}(\nabla_{\boldsymbol{\mu}} \ln p(\boldsymbol{X}, \mathbb{E}_{q_{\boldsymbol{\mu}}}[\boldsymbol{Z}]))|_{\boldsymbol{\mu}=\boldsymbol{\mu}_t}, \tag{2.111}$$

revealing a distinct relationship between solutions to the variational inference problem and their gradients. The paper comes with a negative result as well, though, as this fixed point process turns out to be identical to coordinate ascent updates, so while we shine new light on the venerable method, we fail to capitalize fully on the promise of difference-of-convex optimization, largely owing to the fact that to fully exploit the KL-corrected bound we need to use second-order derivatives, and the convex-concave method is only first order.

The paper *Matrix Product states for inference in discrete probabilistic models*, on the other hand, contributes to the modern black-box approach to variational inference by demonstrating how the discrete REBAR/RELAX estimators can be applied to the novel case of tensor networks, a powerful type of model that we will cover in greater detail in chapter 4. We consider how the model can be reparametrized, explore control variate designs, and build a coupled sampler that allows the model to use the RELAX estimator. We also put a lot of emphasis on how the structured variational approximation contrasts with mean-field models, which can be quite weak approximations for highly structured discrete distributions.

Chapter 3

Gaussian Processes

Gaussian processes are a kind of stochastic process, defined as a set of random variables $f_{\boldsymbol{x}}$ indexed by elements of some domain $\boldsymbol{x} \in \mathcal{X}$ such that any finite subset $\{f_{\boldsymbol{x}_i}\}_{i=1}^{N}$ have a joint Gaussian distribution. This construction can appear very abstract, but the ability to associate a random variable with any point of a domain $\mathcal{X}$ turns out to make Gaussian processes (and other stochastic processes) ideal for modeling functions. Take the simple index space $x \in \mathcal{X} = \mathbb{R}$, for instance. In a standard coordinate system, we could for every x find a random variable f_x corresponding to the value of an unknown random function $f(\cdot)$ at that location. Now imagine a dense grid x_i. Since the f_{x_i} are jointly Gaussian, we can sample them to trace out a random graph (x_i, f_{x_i}). This random graph outlines one possible sample of the random function $f(\cdot)$, and we will henceforth take $f(\boldsymbol{x}) = f_{\boldsymbol{x}}$.

As Gaussians are parameterized by a mean and a covariance, we can imagine collecting the means and covariances for all values $\boldsymbol{x} \in \mathcal{X}$ to extend the parameters to the stochastic process setting. We can take $m(\boldsymbol{x})$ to be the mean function, defining the mean value of each random variable and the random function as a whole, and $k(\boldsymbol{x}, \boldsymbol{x\prime})$ to be the covariance function, yielding the covariance between any two points $\boldsymbol{x}, \boldsymbol{x\prime} \in \mathcal{X}$. The covariance function determines the properties of the random function. If $k(\boldsymbol{x}, \boldsymbol{x\prime})$ is high, the values of $f(\boldsymbol{x})$ and $f(\boldsymbol{x\prime})$ are likely to be similar in value, so if we set the covariance to be high when $\boldsymbol{x}$ is close to $\boldsymbol{x\prime}$ we get more continuous functions, and the distance over which the

covariance decreases is reflected in the distance over which the function changes significantly.

As any finite set of index points induces a finite-dimensional Gaussian distribution, we can consider the joint distribution over two index sets $\boldsymbol{X}_a, \boldsymbol{X}_b \subset \mathcal{X}$ with finite mean and covariance matrix

$$\boldsymbol{m} = \begin{pmatrix} \boldsymbol{m}_a \\ \boldsymbol{m}_b \end{pmatrix}, \quad \boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_a & \boldsymbol{k}_{a,b} \\ \boldsymbol{k}_{b,a} & \boldsymbol{K}_b \end{pmatrix} \tag{3.1}$$

where $[\boldsymbol{m}_a]_i = f(\boldsymbol{x}_i)$ and $[\boldsymbol{K}_a]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}_a$ (similarly for b), and $[\boldsymbol{k}_{a,b}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\boldsymbol{x}_i \in \boldsymbol{X}_a$ and $\boldsymbol{x}_j \in \boldsymbol{X}_b$. In a slight abuse of notation, we will also sometimes write $k(\boldsymbol{X}_a, \boldsymbol{X}_b)$ to mean $\boldsymbol{k}_{a,b}$. Using standard Gaussian calculus, we know that the conditional distribution of $\boldsymbol{X}_b | \boldsymbol{X}_a$ is likewise Gaussian, and has distribution (Rasmussen and C. K. I. Williams 2006)

$$\boldsymbol{m}_{b|a} = \boldsymbol{m}_b + \boldsymbol{k}_{b,a}\boldsymbol{K}_a^{-1}(\boldsymbol{y} - \boldsymbol{m}_a), \quad \boldsymbol{K}_{b|a} = \boldsymbol{K}_b + \boldsymbol{k}_{b,a}\boldsymbol{K}_a^{-1}\boldsymbol{k}_{a,b} \tag{3.2}$$

for observations $y_i = f(\boldsymbol{x}_i)$ for $\boldsymbol{x}_i \in \boldsymbol{X}_a$.

Assuming for the time being that $\boldsymbol{m}_a = \boldsymbol{m}_b = \boldsymbol{0}$, the mean function becomes

$$\boldsymbol{k}_{b,a}\boldsymbol{K}_a^{-1}\boldsymbol{y} \tag{3.3}$$

There are two possible interpretations of this:

Linear Smoother If we group $\boldsymbol{s} = \boldsymbol{K}_a^{-1}\boldsymbol{k}_{b,a}^\top$ we get that the mean function is $\boldsymbol{s}^\top \boldsymbol{y}$, i.e. the predictive mean is a linear smoothing of the observations $\boldsymbol{y}$.

Basis Functions By instead grouping $\boldsymbol{w} = \boldsymbol{K}_a^{-1}\boldsymbol{y}$ we get that the predictive mean is a linear combination of $[\boldsymbol{k}_{b,a}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\boldsymbol{x}_i \in \boldsymbol{X}_b$ and $\boldsymbol{x}_j \in \boldsymbol{X}_a$. We then see that the predictive mean is a linear combination of basis functions $k(\cdot, \boldsymbol{x}_j)$, one for each $\boldsymbol{x}_j$ in the conditioning set.

In addition to the predictive distribution being tractable, we can also compute the marginal evidence,

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, \boldsymbol{K}). \tag{3.4}$$

This is commonly used to tune the Gaussian process model, which can depend on hyperparameters through the mean and covariance functions.

3.1 Kernels

The kernel function essentially defines the Gaussian process. The mean function can matter in inference, but like with ordinary Gaussians we can always restate

a Gaussian process with non-zero mean $f \sim \mathcal{GP}(m, k)$ in terms of a Gaussian process with zero mean $f_0 \sim \mathcal{GP}(0, k)$ as

$$f(\boldsymbol{x}) = m(\boldsymbol{x}) + f_0(\boldsymbol{x}). \tag{3.5}$$

Another reason that people often argue that the mean function does not matter is that it can be subsumed into the kernel. The price is that we have to impose a zero-mean Gaussian scaling coefficient $X \sim \mathcal{N}(0, \lambda)$ on the mean, yielding the augmented random function,

$$\hat{f}(\boldsymbol{x}) = Xm(\boldsymbol{x}) + f(\boldsymbol{x}). \tag{3.6}$$

As we are just adding a Gaussian to a Gaussian for every $\boldsymbol{x}$, the result is another Gaussian process with statistics,

$$\mathbb{E}\left[\hat{f}\right] = 0, \quad \mathbb{E}\left[\hat{f}(\boldsymbol{x})\hat{f}(\boldsymbol{x\prime})\right] = k(\boldsymbol{x}, \boldsymbol{x\prime}) + \lambda m(\boldsymbol{x})m(\boldsymbol{x\prime}) \tag{3.7}$$

which is now zero-mean, but with a new term in the kernel. Of course, this model is different from the fixed mean version; if one wants to model the random fluctuation around some specific known mean behavior, a fixed mean is better as it takes sign and scale into account.

Kernels can be quite different, but an import requirement is that they are always symmetric and positive semi-definite (PSD), which means that for any set $\boldsymbol{X}$ of indices,

$$\boldsymbol{v}^\top k(\boldsymbol{X}, \boldsymbol{X})\boldsymbol{v} \geq 0, \quad \forall \boldsymbol{v} \in \mathbb{R}^{|\boldsymbol{X}|} \tag{3.8}$$

This is equivalent to requiring that $k(\boldsymbol{X}, \boldsymbol{X})$ should always have positive eigenvalues. As an aside, this does not mean that $k(\boldsymbol{X}, \boldsymbol{X})$ has to be element-wise positive, but it does necessitate a positive diagonal; recall that $k(\boldsymbol{X}, \boldsymbol{X})$ describes the covariance of $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \boldsymbol{X}$, so while the off-diagonal covariances can be negative, the on-diagonal variances have to be non-negative. Checking whether a particular function is PSD is a bit of a hassle, so often researchers rely on a more compositional approach. The above derivation of the zero-mean GP illustrated two of the most important rules of what is sometimes called the kernel calculus, which allows for the design of novel kernels by composing existing kernels,

Squaring of a function Any function g can be turned into a kernel by multiplication with itself $k(\boldsymbol{x}, \boldsymbol{x\prime}) = g(\boldsymbol{x})g(\boldsymbol{x\prime})$.

Addition Adding two kernels $k = k_1 + k_2$ yields a new kernel k.

Many more rules like this exist, allowing for the design of rather complex kernels (Bishop 2006; Rasmussen and C. K. I. Williams 2006).

Gaussian processes are intimately related to Bayesian linear regression, which is really just a simpler random function

$$f(\boldsymbol{x}) = \sum_{i=1}^{K} \beta_i \phi_i(\boldsymbol{x}), \quad \beta \sim \mathcal{N}(0, \lambda_i). \tag{3.9}$$

Using the same arguments that we used to subsume the mean function, we can actually convert the Bayesian linear problem into an equivalent Gaussian process with kernel,

$$k(\boldsymbol{x}, \boldsymbol{x\prime}) = \sum_{i=1}^{K} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x\prime}). \tag{3.10}$$

This shows that the classical model is a special case of the Gaussian process, and that standard sets of basis functions map naturally to particular kernels. Linear regression directly on the input features simply corresponds to the kernel,

$$k_{\text{lin}}(\boldsymbol{x}, \boldsymbol{x\prime}) = \boldsymbol{x}^\top \boldsymbol{x\prime}, \tag{3.11}$$

and a quadratic kernel can be found just by squaring the linear kernel (another kernel rule),

$$k_{\text{quad}}(\boldsymbol{x}, \boldsymbol{x\prime}) = (\boldsymbol{x}^\top \boldsymbol{x\prime})^2, \tag{3.12}$$

and we can get a polynomial kernel with all terms of order P simply by doing (Rasmussen and C. K. I. Williams 2006),

$$k_{\text{poly}}(\boldsymbol{x}, \boldsymbol{x\prime}) = (1 + \boldsymbol{x}^\top \boldsymbol{x\prime})^P. \tag{3.13}$$

Note that the kernel k_{poly} summarizes a potentially massive set of basis functions in a very compact fashion, without compromising on expressivity. On the other hand, the fundamental computational bottleneck of Gaussian processes, the inversion of the kernel matrix in equation (3.2), means that it scales poorly in the number of data points N, whereas linear regression scales poorly in the number of features/basis functions. This becomes critical once we move into the fully non-parametric domain, where we allow for an infinite number of basis functions. To see how we can get to an infinite-dimensional set of basis functions, simply take the infinite-order limit of the polynomial kernel to get the exponential kernel:

$$k_{\text{exp}}(\boldsymbol{x}, \boldsymbol{x\prime}) = \lim_{P \to \infty} (1 + \frac{1}{P} \boldsymbol{x}^\top \boldsymbol{x\prime})^P = \exp\big(\boldsymbol{x}^\top \boldsymbol{x\prime}\big). \tag{3.14}$$

Finally, we can get to the most well-known and widely used kernel by using the normalizing kernel construction rule

$$k(\boldsymbol{x}, \boldsymbol{x\prime}) = \frac{k_0(\boldsymbol{x}, \boldsymbol{x\prime})}{\sqrt{k_0(\boldsymbol{x}, \boldsymbol{x})}\sqrt{k_0(\boldsymbol{x\prime}, \boldsymbol{x\prime})}}, \tag{3.15}$$

ensuring that $k(\boldsymbol{x}, \boldsymbol{x}) = 1$. If we think of k as an inner product, this corresponds to calculating the cosine similarity. Applying this normalization to the exponential

kernel, we finally get the squared-exponential (SE) kernel (sometimes called the radial basis function (RBF) or Gaussian kernel as well),

$$k_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x\prime}) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x\prime})^\top(\boldsymbol{x} - \boldsymbol{x\prime})\right). \tag{3.16}$$

Whether it is helpful to think of the SE as a normalized exponential kernel is a matter of taste; an alternative derivation constructs it as the kernel of an infinite number of radial basis functions (Rasmussen and C. K. I. Williams 2006). The SE is widely used because of its tractability, but also because it is infinitely smooth, making it a good kernel for well-behaved functions. The squared-exponential is our first example of a kernel that is only a function of its two inputs through their difference $r = \boldsymbol{x} - \boldsymbol{x\prime}$, making it a member of the class of so-called stationary kernels. Typically, stationary kernels will have two tuning parameters associated to them: the lengthscale ℓ and the prior variance λ, which augments the kernel as

$$k(r) = \lambda k_0\left(\frac{r}{\ell}\right) \rightarrow \quad k_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x\prime}) = \lambda \exp\left(-\frac{1}{2\ell^2}(\boldsymbol{x} - \boldsymbol{x\prime})^\top(\boldsymbol{x} - \boldsymbol{x\prime})\right). \tag{3.17}$$

The prior variance λ we have seen before; when we derived the implicit kernel for a set of basis functions, the coefficient of the corresponding kernel term was exactly the prior variance associated with that basis function. The lengthscale ℓ determines the unit of distance. For an SE kernel, it determines how quickly the exponential drops off, but also implicitly how quickly the function can change. The basis functions of the SE kernel are radial basis functions centered on the indices x_i being conditioned on,

$$k_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x}_i) = \lambda \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{\ell^2}\right), \tag{3.18}$$

so the target function will be approximated by a number of these bump functions — if ℓ is very small, the radial basis functions will be extremely narrow and the function will be able to change more suddenly than if the . Mathematically, we can take the derivative of the basis function:

$$\nabla_{\boldsymbol{x}} k_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x}_i) = -\frac{\lambda}{\ell^2}\exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{\ell^2}\right)(\nabla\|\boldsymbol{x} - \boldsymbol{x}_i\|^2). \tag{3.19}$$

demonstrating that the lengthscale operates as a sort of inverse prior variance for the gradient.

For a stationary kernel, a final property, which is useful both for pedagogical and computational reasons, is the characterization available through Bochner's Theorem (Rahimi and Recht 2008),

Theorem 3.1 (Bochner's) *A continuous kernel $k(\boldsymbol{x}, \boldsymbol{x}\prime) = k_0(\boldsymbol{x} - \boldsymbol{x}\prime)$ on $\mathbb{R}^d$ is positive definite if and only if k_0 is the Fourier transform of a non-negative measure $\Lambda(\omega)$:*

$$k_0(\boldsymbol{x} - \boldsymbol{x}\prime) = \int \exp\bigl(i\boldsymbol{\omega}^\top(\boldsymbol{x} - \boldsymbol{x}\prime)\bigr)\,\mathrm{d}\Lambda(\boldsymbol{\omega})\,. \tag{3.20}$$

This duality means that we can think of stationary kernels in terms of what spatial frequencies their spectral measure $\Lambda(\boldsymbol{\omega})$ is weighting up or down. The SE kernel, for instance, has a Gaussian measure. We can also deduce exactly if a Gaussian process will have Markovian conditional independence structure from the shape of the SE kernel (Hartikainen and Särkkä 2010; Kom Samo and S. J. Roberts 2015). Two other important applications is that we can replace $\Lambda(\boldsymbol{\omega})$ by a sample-based empirical measure $\hat{\Lambda}(\boldsymbol{\omega}) = \sum_{s=1}^{S} \delta_{\omega_s}$, converting a complicated infinite-dimensional kernel into a finite-dimensional kernel or onwards to a set of basis functions known as random Fourier features, allowing efficient finite-dimensional methods to be used (Rahimi and Recht 2008). We can also go the other way, by constructing a rich probability density and then finding the corresponding kernel; using a mixture of Gaussians, the corresponding kernel is known as a spectral mixture kernel (Wilson and Adams 2013). We note that there exists a similar result for non-stationary kernels as well (Kom Samo and S. Roberts 2015).

3.2 Reproducing Kernel Hilbert Spaces

We saw above that the posterior mean function was a linear combination of partially evaluated kernel functions $k(\cdot, \boldsymbol{x}_i)$. We can trivially build a function space containing this class by taking the completion of,

$$\mathcal{H} = \left\{ f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i), \quad n \in \mathbb{N},\ \boldsymbol{x}_i \in \mathcal{X},\ \alpha_i \in \mathbb{R} \right\}, \tag{3.21}$$

and we can assign an inner product to this space by setting

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j\prime), \tag{3.22}$$

for $g(\boldsymbol{x}) = \sum_{i=1}^{m} \beta_j k(\boldsymbol{x}, \boldsymbol{x}_j\prime)$, where the inner product is positive definite by inheriting the property from k. This function space is called a Reproducing Kernel Hilbert Space (RKHS), by virtue of the kernel $k(\cdot, \cdot)$ having the so-called

reproducing property for $f \in \mathcal{H}$,

$$\langle f, k(\cdot, \boldsymbol{x}^*) \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}^*, \boldsymbol{x}_i) = f(\boldsymbol{x}^*). \tag{3.23}$$

The property shows that we can evaluate a function f at $\boldsymbol{x}^*$ by taking the inner product with the reproducing kernel $k(\cdot, \boldsymbol{x}^*)$. This is another way to characterize an RKHS: it is a Hilbert space for which the evaluation functional is bounded, continuous, and is itself a member of the function space. This construction is always possible, as guaranteed by the Moore-Aronszajn theorem which proves that there is an RKHS matching every positive-definite kernel function and vice versa (Aronszajn 1950; Rasmussen and C. K. I. Williams 2006).

To define the Gaussian processes in the context of an RKHS, we need to first recall Mercer's theorem which states that

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y}), \tag{3.24}$$

where ϕ_i is an orthogonal basis of the RKHS with respect to some measure μ, i.e. $\int \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) = \delta_{ij}$. We then resort to the fact that a zero-mean Gaussian process can be written in terms of a similar orthogonal basis as,

$$f(\boldsymbol{x}) = \sum_{i=1}^{\infty} X_i \sqrt{\lambda_i} \phi_i(\boldsymbol{x}), \tag{3.25}$$

where the random variables $X_i \sim \mathcal{N}(0, 1)$ are i.i.d. We see that $\mathbb{E}[f(\boldsymbol{x})] = 0$, while the covariance is

$$\mathbb{E}[f(\boldsymbol{x}) f(\boldsymbol{y})] = \sum_{i,j=1}^{\infty} \mathbb{E}[X_i X_j] \sqrt{\lambda_i \lambda_j} \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}), \tag{3.26}$$

where the last equality is an application of Mercer's theorem, demonstrating that the Gaussian process can be parameterized in terms of a kernel as before. An interesting consequence of this description is that samples from the Gaussian process fall outside the RKHS with probability 1:

$$\mathbb{E}\left[\left\| \sum_{i=1}^{\infty} X_i \lambda_i \phi_i(\boldsymbol{x}) \right\|_{\mathcal{H}}^2 \right] = \sum_{i=1}^{N} \lambda_i \left\langle \phi_i, \phi_i \right\rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} 1 = \infty, \tag{3.27}$$

where we have used that $\langle \phi_i, \phi_j \rangle_{\mathcal{H}} = \delta_{ij} / \lambda_i$. To figure out where the Gaussian process samples reside we can rely on the following theorem (Flaxman et al. 2016),

Theorem 3.2 *Let $\mathcal{H}_k$ be separable. Then $\mathcal{GP}(0, r)$ has sample paths in $\mathcal{H}_k$ if and only if there exists a positive, continuous, self-adjoint, trace-class operator $L : \mathcal{H}_k \to \mathcal{H}_k$ such that*

$$r(\boldsymbol{x}, \boldsymbol{x\prime}) = \langle L[k(\cdot, \boldsymbol{x})], k(\cdot, \boldsymbol{x\prime})\rangle_{\mathcal{H}} \tag{3.28}$$

in which case we say k nuclearly dominates r.

The particular details of this result are outside the scope of this thesis, but we note that an example of such a dominant pair can be established by setting (Flaxman et al. 2016),

$$r(\boldsymbol{x}, \boldsymbol{x\prime}) = \int k(\boldsymbol{x}, \boldsymbol{u}) k(\boldsymbol{u}, \boldsymbol{x\prime}) \, \mathrm{d}\boldsymbol{u} \, . \tag{3.29}$$

A consequence of the posterior mean being in the RKHS is that it inherits the properties of the kernel used, e.g. it has the same degree of smoothness and differentiability as the kernel itself, as it can be written as a finite linear combination of them. Picking a covariance structure implicitly determines a set of basis functions for the posterior mean.

Conversely, we can go from a set of (not necessarily orthogonal) basis functions $\{\varphi_i(\boldsymbol{x})\}_{k=1}^N$ to a covariance matrix, as we saw earlier. This demonstrates that all finite-dimensional function spaces are also RKHS's (Manton and Amblard 2014). In particular, we can associate the span of a single basis vector $\varphi_i(\boldsymbol{x})$ with the rank-1 kernel $k(\boldsymbol{x}, \boldsymbol{x\prime}) = \varphi(\boldsymbol{x})\varphi(\boldsymbol{x\prime})$. Augmenting the set of basis functions corresponds directly to adding rank-1 kernels together. This is formalized as the fact that for RKHS $\mathcal{H}_i$ with reproducing kernel k_i, the kernel $k = \sum_{i=1}^K k_i$ is the reproducing kernel of the RKHS with elements $f = \sum f_i$, $\forall f_i \in \mathcal{H}_i$ (Aronszajn 1950). On a related note, we can use the reproducing kernels to determine whether one RKHS is a subset of another, $\mathcal{H}_{k\prime} \subset \mathcal{H}_k$; if $Mk - k\prime$ is positive definite for some choice of $M \in \mathbb{R}_+$ then $\mathcal{H}_{k\prime}$ is contained in the larger space (Aronszajn 1950).

3.3 Gaussian Process Calculus

An extremely useful property of Gaussian processes is that they are preserved under linear transformations. If we express a Gaussian process in the orthogonal basis form $f(\boldsymbol{x}) = \sum_{i=1}^\infty X_i \sqrt{\lambda_i} \phi_i(\boldsymbol{x})$, then the application of any linear operator A distributes across the orthogonal basis functions

$$Af(\boldsymbol{x}) = \sum_{i=1}^\infty X_i \sqrt{\lambda_i} A\phi_i(\boldsymbol{x}). \tag{3.30}$$

Calculating the second moment of the transformed process, we find that

$$\mathbb{E}[Af(\boldsymbol{x})A\prime f(\boldsymbol{x}\prime)] = \sum_{i=1}^{\infty} \lambda_i A\phi_i(\boldsymbol{x})A\prime\phi_i(\boldsymbol{x}\prime) = A\bar{A}k(\boldsymbol{x},\boldsymbol{x}\prime) \tag{3.31}$$

taking $\bar{A}$ to be the operator A acting on the second input $\boldsymbol{x}\prime$. Since it is zero-mean process, the second moment gives us the covariance function. that This means that not only is the Gaussian process preserved under a linear transform, it is also often tractable — namely in the exact cases where $AA\prime k(\boldsymbol{x},\boldsymbol{x}\prime)$ is tractable. The really interesting feature here is that as the process is derived from the original Gaussian process, the two are highly covariant, with covariance,

$$\mathrm{Cov}(Af(\boldsymbol{x}), f(\boldsymbol{x}\prime)) = \mathbb{E}[Af(\boldsymbol{x})f(\boldsymbol{x}\prime)] = Ak(\boldsymbol{x},\boldsymbol{x}\prime), \tag{3.32}$$

following the same line of proof as above. This means that the two Gaussian processes form a jointly Gaussian system such that for any finite set of indices, the joint distribution is multivariate Gaussian and we can perform joint Gaussian inference. Practically, if $f\prime = Af$, then

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{f}\prime \end{pmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{K}^*\right), \quad \boldsymbol{K}^* = \begin{pmatrix} \boldsymbol{K} & (\boldsymbol{K}\prime)^\top \\ \boldsymbol{K}\prime & \boldsymbol{K}\prime\prime \end{pmatrix} \tag{3.33}$$

with $\boldsymbol{f}$ and $\boldsymbol{f}\prime$ being evaluations of f at $\boldsymbol{X}$ and $f\prime$ at $\boldsymbol{X}\prime$, respectively, and $[\boldsymbol{K}\prime\prime]_{ij} = AA\prime k(\boldsymbol{x}\prime_i, \boldsymbol{x}\prime_j)$, with $\boldsymbol{x}_i\prime, \boldsymbol{x}_j\prime \in \boldsymbol{X}\prime$ and $[\boldsymbol{K}\prime]_{ij} = Ak(\boldsymbol{x}\prime_i, \boldsymbol{x}_j)$, with $\boldsymbol{x}_i\prime \in \boldsymbol{X}\prime, \boldsymbol{x}_j \in \boldsymbol{X}$.

3.3.1 Additive Processes and Linear Algebra

The simplest example of a linear operator is scaling $f\prime = \alpha f$, which translates into a kernel $\alpha^2 k(\boldsymbol{x},\boldsymbol{x}\prime)$. This is a clear parallel to the standard result that $\mathrm{Var}(\alpha X) = \alpha^2 \mathrm{Var}(X)$ for any random variable X. Scaling by a scalar is a property often associated with vector spaces, which leads us to consider whether other properties like addition might also hold for Gaussian processes in a natural fashion. For independent Gaussian processes $f \sim \mathcal{GP}(0, k_f), g \sim \mathcal{GP}(0, k_g)$ it turns out to hold that (Rasmussen and C. K. I. Williams 2006),

$$f + g \sim \mathcal{GP}(0, k_f + k_g), \tag{3.34}$$

Which follows from the identity $\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\,\mathrm{Cov}(X,Y)$; this also immediately tells us how to add Gaussian processes with known covariance. A consequence of these two properties is that we can form linear combinations of Gaussian processes and if we stack multiple independent Gaussian processes into a vector $[\boldsymbol{f}]_i = f_i \sim \mathcal{GP}(0, k_i)$ we can form vector-valued (or

even matrix-valued) Gaussian processes that are amenable to standard linear algebra (Alvarez, Rosasco, and Lawrence 2011). Following the cited survey by Alvarez, Rosasco, and Lawrence, we recognize that the vector-valued $\boldsymbol{f}$ is distributed like a multivariate Gaussian at every index $\boldsymbol{x}$, and that the covariance between two function evaluations is

$$\operatorname{Cov}(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x\prime})) = \begin{pmatrix} k_1(\boldsymbol{x}, \boldsymbol{x\prime}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & k_D(\boldsymbol{x}, \boldsymbol{x\prime}) \end{pmatrix} \equiv \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x\prime}), \tag{3.35}$$

Where $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x\prime})$ is now a matrix-valued kernel, describing the covariances between different dimensions of the vector-valued function. Based on this representation, we can consider what happens when we apply a linear matrix transformation $\boldsymbol{A}\boldsymbol{f}$, which not surprisingly evolves similarly to a standard multivariate Gaussian variable,

$$\boldsymbol{A}\boldsymbol{f} \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{A}\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x\prime})\boldsymbol{A}^\top). \tag{3.36}$$

Returning to the simpler scalar example of (3.34), we note that the relationship is a bit more complicated for the posterior, where we have

$$(f + g)(\boldsymbol{x}^*)|\boldsymbol{y} \sim \mathcal{N}((\boldsymbol{k}_f^* + \boldsymbol{k}_g^*)(\boldsymbol{K}_f + \boldsymbol{K}_g)^{-1}\boldsymbol{y}, \boldsymbol{K}_{f+g|y}) \tag{3.37}$$

$$\boldsymbol{K}_{f+g|y} = \boldsymbol{k}_f(\boldsymbol{x}^*, \boldsymbol{x}^*) + k_g(\boldsymbol{x}^*, \boldsymbol{x}^*) - (\boldsymbol{k}_f^* + \boldsymbol{k}_g^*)^\top(\boldsymbol{K}_f + \boldsymbol{K}_g)^{-1}(\boldsymbol{k}_f^* + \boldsymbol{k}_g^*) \tag{3.38}$$

This almost splits into independent Gaussian processes again, but not quite. Instead of asking for $(f + g)(\boldsymbol{x}^*)|\{(f + g)(\boldsymbol{x}_i), y_i)\}_{i=1}^N$ we can just as well find $(f + g)(\boldsymbol{x}^*)|\{(f + g)(\boldsymbol{x}_i), y_i)\}_{i=1}^N$ using the prior independence to simplify the cross-covariances as

$$\operatorname{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x\prime}) + g(\boldsymbol{x\prime})) = k_f(\boldsymbol{x}, \boldsymbol{x\prime}), \tag{3.39}$$

leaving

$$f(\boldsymbol{x}^*)|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{k}_f^*(\boldsymbol{K}_f + \boldsymbol{K}_g)^{-1}\boldsymbol{y}, k_f(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_f^{*\top}(\boldsymbol{K}_f + \boldsymbol{K}_g)^{-1}\boldsymbol{k}_f^*) \tag{3.40}$$

and similarly for $g(\boldsymbol{x}^*)$. Adding these two partial posteriors together fails to add up to the posterior for the sum. The explanation is of course that by conditioning we have introduced covariance, and using the identity $\operatorname{Var}(X + Y) = \operatorname{Var}(X) + \operatorname{Var}(Y) + 2\operatorname{Cov}(X, Y)$, we can find the covariance by taking the difference between $(f + g)(\boldsymbol{x}^*)|\boldsymbol{y}$ and the two marginal posteriors, leaving

$$\operatorname{Cov}(f(\boldsymbol{x}^*), g(\boldsymbol{x}^*)|\boldsymbol{y}) = -2\boldsymbol{k}_f^{*\top}(\boldsymbol{K}_f + \boldsymbol{K}_g)^{-1}\boldsymbol{k}_g^* \tag{3.41}$$

3.3.2 Bayesian Quadrature

Integrals are perhaps one of the most quintessential examples of a linear operator. The integral lives somewhere in the borderland between tractable and intractable, as many integrals are known and have analytical forms, while others lack solutions. The problem of solving integrals using numerical algorithms has thus been a rich area of research over the years, helped along by the fact that integrals are immensely useful tools, especially due to their intimate relationship with probabilistic expectations. Almost all of the existing numerical algorithms are so-called quadratures, where the integral I is estimated to be a weighted sum of some limited set of function evaluations,

$$I = \int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=1}^{N} w_i f(x_i). \tag{3.42}$$

Some quadrature rules are direct applications of Riemann sums where we assume $x_i < x_{i+1}$ and take $w_i = |x_{i+1} - x_i|$ and $w_N = 0$, estimating the areas of rectangles in the intervals between observations. Variations like the trapezoid rule use trapezoids in place of rectangles, and we could probably imagine other shapes too, but the approaches have the feature in common that they are often formulated in terms of estimating local areas. A more natural point of view, though, might be to think in terms of approximating the integrand directly, and then taking the integral of the approximation; we just have to pick an approximation where the integral is tractable. Both of the two described approaches can be defined naturally in function terms: the Riemann sum tries to approximate the integrand as piecewise constant, while the trapezoid rule uses a piecewise linear approximation. From this, it also becomes clear that both quadrature rules are making strong assumptions about the continuity, differentiability, and smoothness of the underlying integrand. If the function was allowed to vary wildly and discontinuously inside the intervals, even these conservative estimates would be poor fits. Conversely, if the function is infinitely smooth, the two quadratures are not efficient approximations, eschewing information that could be used to make higher fidelity approximations.

The idea of most numerical algorithms to use limited information to make predictions about unknown quantities is fundamentally a problem of inference. The field of probabilistic numerics has dedicated itself to this point of view, demonstrating that many classical numerical problems can be modeled probabilistically, and that many classical numerical algorithms reduce to MAP solutions of these models (Cockayne et al. 2017). Bayesian quadrature is the probabilistic numerics equivalent of normal quadrature, where we approximate the integrand using a Gaussian process, encoding our assumptions into the kernel (Briol et al. 2015). The quadrature itself follows by applying the integral operator $\Pi[\cdot]$ to

the Gaussian process (Briol et al. 2015),

$$f \sim \mathcal{GP}(0, k) \Rightarrow \Pi[f] \sim \mathcal{N}\left(0, \Pi\Pi[k(\cdot,\cdot)]\right) \tag{3.43}$$

with $\Pi\Pi[k(\cdot,\cdot)]$ denoting the application of the operator to both arguments of k. Since the integral operator is a functional, mapping from a function space to the reals, we find that the Gaussian process reduces to a Gaussian. With everything jointly Gaussian, we can easily find the posterior on the integral to be,

$$\Pi[f] \,|\, \boldsymbol{f} \sim \mathcal{N}\left(\Pi[\boldsymbol{k}(\cdot,\boldsymbol{X})]\boldsymbol{K}^{-1}\boldsymbol{f},\ \Pi\Pi[k(\cdot,\cdot)] - \Pi[\boldsymbol{k}(\cdot,\boldsymbol{X})]\boldsymbol{K}^{-1}\Pi[\boldsymbol{k}(\boldsymbol{X},\cdot)]\right) \tag{3.44}$$

Focusing on the posterior mean, we recognize it as a quadrature with weights,

$$w_i = [\Pi[\boldsymbol{k}(\cdot,\boldsymbol{X})]\boldsymbol{K}^{-1}]_i. \tag{3.45}$$

Bayesian quadrature offers a number of advantages, such as a natural uncertainty estimate, more direct control about the function assumptions through the kernel, and the ability to include integrals in more complex probabilistic model, a simple example being to include input noise in the model, allowing approximate integration of noisy functions.

If we take several integrals, say $\Pi_1[f]$ and $\Pi_2[f]$ they will of course also be correlated,

$$\mathrm{Cov}(\Pi_1[f], \Pi_2[f]) = \langle \Pi_1[k(\cdot,\cdot)], \Pi_2[k(\cdot,\cdot)] \rangle_{\mathcal{H}}\,. \tag{3.46}$$

From this equation, we get that the covariance is determined by the inner product of functions $\mu_\Pi \in \mathcal{H}$,

$$\mu_\Pi(\boldsymbol{x}) = \Pi[k(\cdot,\boldsymbol{x})]. \tag{3.47}$$

This function summarizes the integral, allowing us to compute all covariances in terms of inner products with this special function e.g. $\mathrm{Cov}(\Pi[f], f(\boldsymbol{x})) = \langle \mu, k(\cdot,\boldsymbol{x}) \rangle$. Additionally, just like $k(\cdot,\boldsymbol{x})$ has the reproducing property for function evaluation, $\mu(\cdot)$ reproduces the integral as

$$\langle \mu_\Pi, f \rangle_{\mathcal{H}} = \sum_{i=1}^{N} \alpha_i \langle \mu_\Pi, k(\cdot,\boldsymbol{x}_i) \rangle_{\mathcal{H}} = \sum_{i=1}^{N} \alpha_i \Pi[k(\cdot,\boldsymbol{x}_i)] = \Pi[f]. \tag{3.48}$$

This is related to the functional analysis concept of a Riesz representation, hinging on the famous Riesz representation theorem which states that any functional F on a Hilbert space $\mathcal{H}$ can be represented as $Af = \langle f, v_A \rangle$ for some unique representer $v_A \in \mathcal{H}$ (Kreyszig 1978).

We call μ_Π an RKHS embedding of the operator Π. When Π is an expectation, that is, of the form

$$\Pi[f] = \int p(\boldsymbol{x}) f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}\,, \tag{3.49}$$

for a probability density function $p(\boldsymbol{x})$, then the embedding is called the kernel mean embedding and is denoted μ_p or μ_X if p is associated with a random variable X. One consequence of moving to the probabilistic domain is that we can naturally restate the inner product as an expectation,

$$\langle \mu_p, \mu_q \rangle_{\mathcal{H}} = \mathbb{E}_{\boldsymbol{x}\sim p, \boldsymbol{x}\prime\sim q}[k(\boldsymbol{x}, \boldsymbol{x}\prime)]\,, \tag{3.50}$$

lending itself to the interpretation that we are measuring the similarity of p and q by considering how similar their samples are on average.

Although the kernel mean embedding is just a Riesz representation of the expectation by the above argument, it turns out that it is a very strong representation of the underlying distribution as well. In particular, if we embed the probability measure into a universal RKHS, the map is injective, and each possible embedding then corresponds to a specific probability measure (Smola et al. 2007; Muandet et al. 2016). This property of inducing injective embeddings is also possessed by some non-universal RKHSs, and spaces for which it holds are referred to as characteristic (Sriperumbudur, Fukumizu, and Lanckriet 2011). That a distribution can be distinguished by its expectations alone might initially seem a bit peculiar, but relates to the venerable concept of moment-generating functions,

$$M_{\boldsymbol{X}}(\boldsymbol{t}) = \mathbb{E}\big[\exp\big(\boldsymbol{t}^\top \boldsymbol{X}\big)\big], \tag{3.51}$$

which are known to uniquely characterize any probability distribution for which it is defined. Moment-generating functions manage to summarize the qualities of a distribution in terms of a its expectation over a simple test function. Noticeably, the moment-generating function appears to be a kernel mean embedding itself, with the exponential kernel $\exp\big(\boldsymbol{x}^\top \boldsymbol{x}\prime\big)$, that we used earlier to generate the SE kernel, as its embedding kernel. A similar argument, originally due to Muandet et al. (2016), shows that by applying Bochner's theorem to the expression in equation 3.50,

$$\langle \mu_p, \mu_q \rangle_{\mathcal{H}} = \iint p(\boldsymbol{x}) q(\boldsymbol{x}\prime) k(\boldsymbol{x}, \boldsymbol{x}\prime)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{x}\prime = \tag{3.52}$$

$$\iint p(\boldsymbol{x}) q(\boldsymbol{x}\prime) \left[\int \exp\big(i\boldsymbol{\omega}^\top(\boldsymbol{x} - \boldsymbol{x}\prime)\big)\, \mathrm{d}\Lambda(\boldsymbol{\omega})\right] \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{x}\prime = \tag{3.53}$$

$$\int \left(\int p(\boldsymbol{x}) \exp\big(i\boldsymbol{\omega}^\top \boldsymbol{x}\big)\, \mathrm{d}\boldsymbol{x}\right) \left(\int q(\boldsymbol{x}\prime) \exp\big(i\boldsymbol{\omega}^\top \boldsymbol{x}\prime\big)\, \mathrm{d}\boldsymbol{x}\prime\right) \mathrm{d}\Lambda(\boldsymbol{\omega}) = \langle \phi_p, \phi_q \rangle_{L^2_\Lambda} \tag{3.54}$$

where $\phi_p(\boldsymbol{x}) = \int p(\boldsymbol{x}) \exp\big(i\boldsymbol{\omega}^\top \boldsymbol{x}\big)\, \mathrm{d}\boldsymbol{x}$ is the characteristic function, the generalized version of moment-generating functions. So no matter what stationary kernel we employ, the inner product in L^2 is always with respect to the two characteristic functions, but with the spectral density $\Lambda(\boldsymbol{\omega})$ of the kernel weighting the different frequencies.

3.4 Variational Inducing Point Methods

Gaussian processes are notorious for their computational demands. Referring back to the posterior mean and covariance of equation (3.2), we note that there is a computational bottleneck in the form of the matrix inverse, which costs $\mathcal{O}(N^3)$. Instead of computing the full posterior, we can take a variational approach and find a computationally efficient approximation.

A challenge particular to variational approximations for Gaussian processes is that the posterior is over a potentially infinite number of points, as we can extend the posterior to any element of the domain. Since the cost is cubic in the number of points we are conditioning on, some older methods speculated that we could compress the posterior by conditioning on a smaller subset of points, eliminating redundancies in the training set without compromising the posterior too much. This idea was later carried over to the variational paradigm (Titsias 2009).

The setup is that we assume a vector of N real observations $\boldsymbol{y}$ made at a set of locations $\boldsymbol{X}$. We assume that these observations are related to a vector $\boldsymbol{f}$ of N latent evaluations of a Gaussian process such that $f_i = f(x_i), \forall x_i \in \boldsymbol{X}$ and $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$. In addition to this, we augment our model with a set of latent evaluations $\boldsymbol{f}_u$ from the GP at an auxiliary set of locations $\boldsymbol{U}$, which we will denote the inducing points. We note that $\boldsymbol{f}$ and $\boldsymbol{f}_u$ are strongly dependent on each other through the GP prior. Using Bayes', the posterior of $\boldsymbol{f}$ and $\boldsymbol{f}_u$ can trivially be stated as,

$$p(\boldsymbol{f}, \boldsymbol{f}_u|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{f}_u)p(\boldsymbol{f}_u). \tag{3.55}$$

Note that we assume here that the likelihood is conditionally independent of $\boldsymbol{f}_u$ given $\boldsymbol{f}$. Variationally, we could fit $p(\boldsymbol{f}, \boldsymbol{f}_u|y)$ with a Gaussian approximation, but unless the covariance matrix is sufficiently structured to reduce the computational burden, it would not do us much good. Instead, we can apply a full Gaussian approximation only to the subset of $p(\boldsymbol{f}_u)$, and let $\boldsymbol{f}$ be modeled directly through the conditional relationship by assuming that our variational model has the form

$$q(\boldsymbol{f}, \boldsymbol{f}_u) = p(\boldsymbol{f}|\boldsymbol{f}_u)q(\boldsymbol{f}_u) \tag{3.56}$$

where we emphasize that p is the true conditional relationship. Since the likelihood is a function of $\boldsymbol{f}$, this forces $q(\boldsymbol{f}_u)$ to model latent observations that *induce* an appropriate posterior on $\boldsymbol{f}$ to fit the data, explaining the name. Mathematically, there is a convenient bit of cancellation in the ELBO from our assumptions

$$\mathcal{L} = \mathbb{E}\left[\ln \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{f}_u)p(\boldsymbol{f}_u)}{p(\boldsymbol{f}|\boldsymbol{f}_u)q(\boldsymbol{f}_u)}\right] = \mathbb{E}_{q(\boldsymbol{f})}[\ln p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}\left(q(\boldsymbol{f}_u)\|p(\boldsymbol{f}_u)\right). \tag{3.57}$$

This splits into the usual expected log-likelihood and prior divergence decomposition, but the expectation of the likelihood is now with respect to the marginalized distribution $q(\boldsymbol{f}) = \int p(\boldsymbol{f}|\boldsymbol{f}_u)p(\boldsymbol{f}_u)\,\mathrm{d}\boldsymbol{f}_u$. Using a Gaussian distribution $q(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$ for the approximation makes the KL divergence tractable, and we can also compute the marginalized density,

$$q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{k}_{xu}\boldsymbol{K}_{uu}^{-1}\boldsymbol{f}_u, \boldsymbol{K}_{xx} - \boldsymbol{k}_{xu}\boldsymbol{K}_{uu}^{-1}\boldsymbol{k}_{xu}^\top)\,\mathcal{N}(\boldsymbol{f}_u|\boldsymbol{\mu}, \boldsymbol{S})\,\mathrm{d}\boldsymbol{f}_u = \tag{3.58}$$

$$\mathcal{N}(\boldsymbol{k}_{xu}\boldsymbol{K}_{uu}^{-1}\boldsymbol{\mu}, \boldsymbol{K}_{xx} + \boldsymbol{k}_{xu}\boldsymbol{K}_{uu}^{-1}(\boldsymbol{S} - \boldsymbol{K}_{uu})\boldsymbol{K}_{uu}^{-1}\boldsymbol{k}_{xu}^\top), \tag{3.59}$$

using subscripts to denote whether the kernel is over $\boldsymbol{X}$, $\boldsymbol{U}$, or a mix. To perform the integral we used standard Gaussian identities (Bishop 2006; Damianou and Lawrence 2012). For notational convenience, we will denote $q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\tilde{\boldsymbol{m}}, \tilde{\boldsymbol{K}})$.

A related bound can be derived by implementing the KL-corrected bound, taking $\boldsymbol{f}_u$ to be the nuisance variable. The auxiliary bound $\mathcal{L}_1 \leq \ln p(\boldsymbol{y}|\boldsymbol{f}_u)$ is,

$$\mathcal{L}_1(\boldsymbol{f}_u) = \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{f}_u)}\left[\ln \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{f}_u)}{p(\boldsymbol{f}|\boldsymbol{f}_u)}\right] = \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{f}_u)}[\ln p(\boldsymbol{y}|\boldsymbol{f})] \tag{3.60}$$

which would normally depend on $q(\boldsymbol{f}|\boldsymbol{f}_u)$, but we follow the inducing input derivation and set the approximation to the prior conditional $p(\boldsymbol{f}|\boldsymbol{f}_u)$. The KL-corrected bound is then

$$\mathcal{L}_{KL} = \ln \mathbb{E}_{p(\boldsymbol{f}_u)}[\exp(\mathcal{L}_1(\boldsymbol{f}_u))] \tag{3.61}$$

In the case of a Gaussian likelihood $\mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2\boldsymbol{I})$, the two bounds become

$$\mathcal{L}_1(\boldsymbol{f}_u) = \ln \mathcal{N}(\boldsymbol{y}|\boldsymbol{m}_{f|f_u}, \sigma^2\boldsymbol{I}) - \frac{1}{2}\operatorname{Tr}[\boldsymbol{K}_{f|f_u}] \tag{3.62}$$

$$\mathcal{L}_{KL} = \ln \mathcal{N}(\boldsymbol{y}) - \frac{1}{2\sigma^2}\operatorname{Tr}[\boldsymbol{K}_{f|f_u}] \tag{3.63}$$

where $\boldsymbol{m}_{f|f_u}$ and $\boldsymbol{K}_{f|f_u}$ is the mean and covariance of the standard predictive posterior, respectively, of $p(\boldsymbol{f}|\boldsymbol{f}_u)$ while $\bar{\boldsymbol{K}}$ is the covariance of the marginal,

$$\int \mathcal{N}(\boldsymbol{y}|\boldsymbol{m}_{f|f_u}, \sigma^2\boldsymbol{I})\,\mathcal{N}(\boldsymbol{f}_u|\boldsymbol{0}, \boldsymbol{K}_{uu})\,\mathrm{d}\boldsymbol{f}_u = \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I} + \boldsymbol{k}_{fu}\boldsymbol{K}_{uu}^{-1}\boldsymbol{k}_{fu}^\top) \tag{3.64}$$

This cost function does not depend on $q(\boldsymbol{f}_u)$, but still depends on any kernel parameters, making it a good proxy for empirical Bayes optimization of the hyperparameters, especially due to the ELBOs lower-bounding property. This is also Titsias' original derivation (Titsias 2009), while other sources discuss the relationship to the more general bounds in greather detail (Damianou 2015). Using the properties of the KL-corrected bound, we can additionally retrieve the optimal $q^*(\boldsymbol{f}_u)$ at any time as (Hensman, Rattray, and Lawrence 2012),

$$q(\boldsymbol{f}_u)^* \propto p(\boldsymbol{f}_u)\exp(\mathcal{L}_1(\boldsymbol{f}_u)) \propto \mathcal{N}(\boldsymbol{y}|\boldsymbol{m}_{f|f_u}, \sigma^2\boldsymbol{I})\,\mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{uu}) \tag{3.65}$$

which we integrated above, simultaneously showing $\mathcal{L}_{KL}$ to be the normalization constant. Another feature of the KL-corrected bound is that it can be derived by finding q^* first, and then plugging it into the original inducing point ELBO of equation (3.57),

$$\mathcal{L}_{KL} \equiv \mathbb{E}_{q^*}[\ln p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}\left(q^*(\boldsymbol{f}_u)\|p(\boldsymbol{f}_u)\right). \tag{3.66}$$

A similar derivation, specifically in the context of Gaussian processes, made by King and Lawrence (2006) was the direct inspiration for the general KL-correction procedure.

Two issues with this fully marginalized bound is that it does not apply to non-conjugate likelihoods and that it does not decompose over individual observations y_i, making it difficult to scale to large datasets via e.g. SVI. In these cases, we will have to return to the standard inducing points objective of equation (3.57) where we note that for any factorizing likelihood $p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} p(y_i|f_i)$

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q(f_i)}[\ln p(y_i|f_i)] - \mathrm{KL}\left(q(\boldsymbol{f}_u)\|p(\boldsymbol{f}_u)\right) \tag{3.67}$$

where we remark that the expectation is only with respect to univariate marginals $q(f_i)$. This means we can apply standard quadrature methods to approximate the integral with good accuracy (Hensman, Matthews, and Ghahramani 2015).

3.5 Contribution

Our paper *Joint expectation kernels* relates directly to Gaussian processes, by proposing the joint expectation kernel as a generalization of the natural kernel between kernel mean embeddings. As we showed in equation (3.50), the natural kernel on distributions can be written directly as the inner product between the kernel mean embeddings of the integrals like,

$$\mathrm{Cov}_f(\mathbb{E}_p[f], \mathbb{E}_q[f]) = \mathbb{E}_{\boldsymbol{x}\sim p, \boldsymbol{x}\prime\sim q}[k(\boldsymbol{x}, \boldsymbol{x}\prime)] = \langle \mu_p, \mu_q \rangle_{\mathcal{H}}, \quad \mu_p = \int p(\boldsymbol{x}) k(\boldsymbol{x}, \cdot)\, \mathrm{d}\boldsymbol{x}.$$

As we noted earlier, an attractive interpretation follows from the expectation of the above equation, as it seems to state that similarity of distributions is the average kernel similarity of random draws. This seems reasonable when we are comparing independent distributions, and if we compare a distribution to itself, we get the kernel

$$\mathbb{E}_{\boldsymbol{x}, \boldsymbol{x}\prime\sim p}[k(\boldsymbol{x}, \boldsymbol{x}\prime)], \tag{3.68}$$

where $\boldsymbol{x}$ and $\boldsymbol{x\prime}$ are i.i.d. draws from p. This is a kernel directly on distributions, as we note that the kernel does not distinguish between comparing p to itself, or to a copy $p\prime$ with the exact same distribution. But what if we want to distinguish between the two? In particular, we might imagine a kernel on random variables $X \sim p$, which continue to be distributed according to particular marginal distributions, but share a joint sample space. What we propose in our papers, is to use the kernel

$$k_E(X, X\prime) = \mathbb{E}_{X,X\prime \sim p_{XX\prime}}[k(X, X\prime)]\,, \tag{3.69}$$

which is completely identical to the kernel on kernel mean embeddings, except that we are taking the expectation with respect to the joint distribution now. We show that this is a proper kernel, and we contrast it with the standard kernel, noting a number of unintuitive consequences. Interestingly, even if we only consider independent random variables, the kernels are still different, as every X is trivially dependent with respect to itself, leaving the diagonal of the kernel matrix different:

$$k_E(X, X) = \mathbb{E}[k(X, X)] = k(0). \tag{3.70}$$

We note how this can be used to handle censored data in a simple fashion, as an approximate method for noisy input data, or to embed entire Gaussian processes.

Chapter 4

Tensor Networks

Conditional independence is the key to most successful probabilistic machine learning algorithms, but as we saw with Gaussian processes, sometimes complete co-dependence is a necessary component of the inference. Conditional independence assumptions are even more common in discrete probabilistic models, and the graphical models literature is rife with algorithms running on trees and DAGs. Nevertheless, it is easy to construct natural models where everything is dependent. Any model where the discrete variables model a labeling or partitioning of a set of objects is particularly susceptible to co-dependence. Take clustering, for instance; any point added to a cluster will influence whether all other points are likely to be put in the same cluster or not. The practical problem we are facing is one of scale: if there are N variables with K different discrete states we have to model K^N different configurations, which quickly becomes an insurmountable number.

We try to attack this seemingly impossible problem by restating it as a tensor approximation problem and using a low-rank assumption to make it amenable.

This will be a shorter chapter, as the associated paper is already rather expansive.

4.1 Discrete Probabilistic Models as Tensors

We let $\boldsymbol{X}$ be a set of N discrete variables, with $x_n \in \boldsymbol{X}$ taking values in the discrete set $\mathcal{X}_n$ of cardinality $K_n = |\mathcal{X}_n|$ and we let $\boldsymbol{Y}$ be an observed set of continuous and/or discrete random variables, jointly distributed with $\boldsymbol{X}$ under the probabilistic model $p(\boldsymbol{X}, \boldsymbol{Y})$.

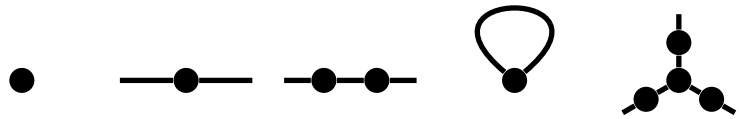
Conditioned on $\boldsymbol{Y}$, the posterior is a discrete distribution with $\prod_{n=1}^{N} K_n$ states. If $K_n = K$ for all n, then there are K^N states, which quickly leads to a combinatorial explosion for even small K and N. The challenge, as always, is the normalization constant for the posterior (or any marginal thereof). Moving to discrete models means that we no longer have to contend with integrals lacking closed form solutions, as every marginalization is just a sum. But if there is K^N elements in the sum, we will quickly reach the limits of what is tractable. Conditional independence helps by breaking the large sum into a series of sub-sums,

$$\sum_{x_1,x_2} p(y, x_1, x_2) = \sum_{x_1,x_2} p(y|x_2)p(x_2|x_1) = \sum_{x_2} p(y|x_2) \sum_{x_1} p(x_2|x_1) \tag{4.1}$$

So instead of one sum with $K_1 K_2$ terms, we get to compute two sums in sequence, with K_1 and K_2 terms, respectively. This is the idea exploited by most discrete graphical model inference algorithms, in particular message-passing methods and the junction-tree algorithm (Wainwright and Jordan 2008). This is a direct parallel to what we saw for continuous models in the introduction.

If our model lacks conditional independence structure, these algorithms fall short. Taking a step away from graphical models, another way to represent the posterior is as any mathematical structure that can contain the posterior probabilities associated with each of the K^N possible configurations.

A tensor is an array $\mathcal{T}$ which can have up N dimensions or modes, each of length K_n. It can be indexed with a set of multidimensional indices $\mathcal{I} = (i_1, \ldots, i_N)$ with $i_n \in 1, \ldots, K_n$ where we here assume that $\mathcal{T}_\mathcal{I} \in \mathbb{R}$ so that $\mathcal{T} \in \mathbb{R}^{K_1 \times \ldots \times K_N}$. From this, it should be clear that we can associate a tensor with any discrete probability distribution: by assuming the discrete sample space to be $\mathcal{X}_n = \{1, \ldots, K_n\}$ without loss of generality, we can associate an index $\mathcal{I}$ with a configuration $x_n = i_n, \forall n$, letting us tie all values $p(\boldsymbol{X} = \mathcal{I}|\boldsymbol{Y})$ to an element in a tensor $\mathcal{T}_\mathcal{I}$ with one dimension for each of the N random variables, and each dimension having length K_n, corresponding to each discrete state of x_n.



(a) A scalar. **(b)** A matrix. **(c)** A matrix product. **(d)** A matrix trace. **(e)** Tucker decomposition.

Figure 4.1: Different expressions in the graphical language of tensor networks.

4.2 Tensor Networks

A tensor network expresses a tensor in terms of smaller independent tensors and index contractions. When $N = 2$, a tensor is just a matrix, so we will use a simple example to illustrate the idea. Consider two matrices A_{ij} and B_{jk}. We can take the so-called tensor product of these two matrices to get a new 3-dimensional tensor by tying the second dimension of A and the first dimension of B,

$$M_{ijk} = A_{ij}B_{jk} \tag{4.2}$$

This is already a simple example of a tensor network, but we also allow contractions over indices, where we sum over all possible values of for example index j:

$$M_{ik} = \sum_j A_{ij}B_{jk} \tag{4.3}$$

This is just another way to write $\boldsymbol{M} = \boldsymbol{AB}$ so if j only ranges over a few values, we recognize this as a low-rank factorization which is a well-known method for simplifying complicated matrices and matrix expressions.

In the tensor network literature, a graphical language is often used to express models. The tensor network is expressed as a graph, where nodes are tensors and edges indicate the indices of each tensor. So a scalar will have no edges, a vector a single edge, a matrix two edges, and so on. Dangling edges that extend from one node but do not connect to another indicate the open indices of the complete tensor represented by the full network, i.e. the edges corresponding to indices i and k in the low-rank method would be dangling. Edges connecting two nodes represent contractions, and the indices of the involved tensors are matched and summed over. See figure 4.1 for examples. Given a graph with N dangling edges, we can index by contracting the dangling indices with a monomial unit vector $\boldsymbol{e}_{i_n}$, until all dangling edges are closed off. This leaves us with one or more scalars graphs, that are implicitly assumed to be multiplied together.

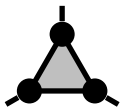


Figure 4.2: CP decomposition with hyperedge.

Many classical tensor decomposition algorithms are easily restated in this graphical language, for example the tucker decomposition,

$$\mathcal{T}_{ijk} = \sum_{r,s,t} \mathcal{G}_{rst} U_{ri} U_{sj} U_{tk} \tag{4.4}$$

which can be identified as a tensor network as the formula only involves subtensors and contraction operations, and we can thus write it graphically as in figure 4.1e.

To extend the graphical language to contractions where more than two tensors share an index, it becomes necessary to allow hyperedges in the tensor network. A hyperedge is an edge that can connect more than two vertices (tensors), so just as a standard edge can be represented by the set $\{v_1, v_2\}$ of its two adjoining vertices, a hyperedge is just a set of greater cardinality $\{v_1, \ldots, v_K\}$. In the drawings, we can represent hyperedges as shaded triangles. Ironically, one model where hyperedges are convenient is for the CP decomposition, visualized in figure 4.2, often considered a simpler version of the Tucker decomposition (Kolda and Bader 2009). It should be noted that it is not strictly necessary to use hyperedges, though; instead, we can use auxiliary tensors such as $\mathcal{A}_{ijk} = \mathbb{1}[i = j = k]$ for three edges, where the indicator function is one when the logical statement is true.

4.3 Relationship with Graphical Models

The graphical language of tensor networks is superficially similar to probabilistic graphical models, and in fact the relationship goes even deeper. To explain, we provide the following definition of an undirected graphical model from Robeva and Seigal (2017). For a hypergraph H with hyperedge set $\mathcal{C}$, an undirected graphical model with respect to H is any probabilistic model that factors as

$$p(\boldsymbol{X}) = \frac{1}{\mathcal{Z}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{X}_C), \tag{4.5}$$

where ψ_C is a clique potential for clique C: some function over $\boldsymbol{X}_C$ which is the configuration $\boldsymbol{X}$ on the subset of variables that are members of the clique. For

proper distributions these functions should be non-negative, but we relax that for the time being. The graphical model in question then has a node for each random variable, and a hyperedge for each clique. This definition leads us to the following duality theorem, presented in a recent article (Robeva and Seigal 2017, Theorem 2.1),

Theorem 4.1 *A discrete graphical model associated to a hypergraph H with clique potentials $\psi_C : \prod_{u\in C} \mathcal{X}_u \to \mathbb{R}$ is the same as the data of a tensor network associated to the dual hypergraph H^* with tensors $\mathcal{T}_C = \psi_C$ at each vertex of H^*.*

This duality might appear a bit esoteric, but this means a lot of results carry over from graphical models to tensor networks. We will show how this can be applied to a particular branch of efficient tensor networks.

4.4 Tensor Trains and Rings

Most of the literature on tensor networks comes from quantum mechanics, but a few particular forms of tensor network have started to diffuse over into the tensor factorization and general machine learning literatures. One such example is the tensor train (Oseledets 2011), which can be most easily summarized by using the graphical notation as in figure 4.3a. The tensor train (TT) is built up of N unique 3-tensors $\mathcal{G}_n \in \mathbb{R}^{r_{n-1}\times K\times r_n}$ which we call the cores, where we note n is a non-tensorial index as the other indices change along with it. The $\{r_n\}_{n=1}^N$ are the tensor train ranks (TT-ranks) of the model, and describes the capacity of the tensor train, in the same sense as for a common low-rank factorization. To simplify the notation a bit, we will write $\mathcal{G}_n[i_n] \in \mathbb{R}^{r_{n-1}\times r_n}$ to denote the core matrix we get when n and i_n is fixed. Using this, we can write

$$\mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1]\mathcal{G}_2[i_2]\dots\mathcal{G}_N[i_N]. \tag{4.6}$$

Note that since this needs to collapse to a scalar, we need to require that $r_0 = r_N = 1$.

We can also wrap the graph around itself, giving us the circular structure in 4.3b, sans the dotted edge. This makes it easier to see the similarity to the Tucker model. Adding the dotted edge gives us the aptly named tensor ring, which modifies the tensor train with a trace operator (Zhao et al. 2016),

$$\mathcal{T}_{\mathcal{I}} = \mathrm{Tr}[\mathcal{G}_1[i_1]\mathcal{G}_2[i_2]\dots\mathcal{G}_N[i_N]]. \tag{4.7}$$

An advantage is that this allows $r_0 = r_N$ to take on values different from 1, and on the design side we do not need to worry about the cyclic ordering of the cores.

It's also more expressive, as we can expand the trace into a sum of r_0 tensor trains with different first and last cores.

As opposed to e.g. the Tucker model, matrix product states scale well in the number of dimensions. We need to add another matrix for each dimension, but never need to take a tensor of order higher than 3 into account, preventing combinatorial explosions. A further advantage is that there exists an algorithm for computing the exact tensor train of any tensor (Oseledets 2011). Unfortunately, it scales poorly, as it involves SVDs of matrix unfoldings of the tensor. Still, the constructive algorithm shows that the tensor train is powerful enough to capture any tensor, given sufficient rank. These qualities make tensor trains well-suited for approximating the large tensors we are facing, but two critical issues remain: how do we ensure positivity of the tensor elements, and how do we calculate normalization.

To ensure positivity, we could trivially set all the core tensors to be positive, but it is unclear whether this would compromise the model's capacity. Instead, we will borrow a trick from the quantum mechanics literature, where matrix product states are used to model wave functions that can be squared to yield probability distributions via Born's rule. In a bit of notational disagreement, matrix product states is an umbrella term for both tensor trains and tensor rings in the quantum literature, although they distinguish between the two by stating that the tensor train has an open boundary condition and the tensor ring a periodic boundary condition (Schollwöck 2011). We will reserve the MPS label for the specific setting of a squared tensor train (or ring) modeling a probability density,

$$p(\boldsymbol{X} = \mathcal{I}) = (\mathcal{T}_{\mathcal{I}})^2 \tag{4.8}$$

This solves the positivity problem, leaving normalization. For the tensor train, we can expand the square as,

$$(\mathcal{T}_{\mathcal{I}})^2 = \mathcal{G}_N[i_N]^\top \dots \mathcal{G}_2[i_2]^\top \mathcal{G}_1[i_1]^\top \mathcal{G}_1[i_1] \mathcal{G}_2[i_2] \dots \mathcal{G}_N[i_N] \Rightarrow \tag{4.9}$$

$$\mathcal{Z} = \sum_{\mathcal{I}} (\mathcal{T}_{\mathcal{I}})^2 = \sum_{i_N} \mathcal{G}_N[i_N]^\top \dots \sum_{i_2} \mathcal{G}_2[i_2]^\top \left(\sum_{i_1} \mathcal{G}_1[i_1]^\top \mathcal{G}_1[i_1] \right) \mathcal{G}_2[i_2] \dots \mathcal{G}_N[i_N] \tag{4.10}$$

A key quantity here is the recursively defined set of left marginals,

$$\boldsymbol{L}_n = \sum_{i_n} \mathcal{G}_n[i_n]^\top \boldsymbol{L}_{n-1} \mathcal{G}_n[i_n], \quad \boldsymbol{L}_0 = \boldsymbol{I}. \tag{4.11}$$

Alternatively, we could have summed out i_N first and i_1, giving us a recursion in,

$$\boldsymbol{R}_n = \sum_{i_n} \mathcal{G}_n[i_n] \boldsymbol{L}_{n+1} \mathcal{G}_n[i_n]^\top, \quad \boldsymbol{R}_{N+1} = \boldsymbol{I}. \tag{4.12}$$

with both $\boldsymbol{R}_1 = \boldsymbol{L}_N = \mathcal{Z}$. So, like with the conditional independence property we demonstrated in the introduction to the chapter, we can distribute the sum over K^N elements in such a way that we can compute it via N sums of length K. Naturally, we can use the same principle to compute any marginal:

$$p(x_n = i_n) \propto \sum_{\mathcal{I}_{-n}} (\mathcal{T}_{\mathcal{I}})^2 = \mathrm{Tr}[\mathcal{G}_n[i_n]^\top \boldsymbol{L}_{n-1} \mathcal{G}_n[i_n] \boldsymbol{R}_{n+1}]. \tag{4.13}$$

Here we use to $\mathcal{I}_{-n}$ to denote the set of indices excluding i_n. This extend to multivariate marginals when the variables of interest occur in consecutive order, but for arbitrary marginals we will have to sum out some of the variables explicitly, making some marginals more expensive than others.

4.4.1 Canonical Cores

Another result from quantum mechanics is that there exists so-called canonical sets of cores.As it is, any set of cores $\mathcal{G}_n$ can be replaced via a so-called gauge transform,

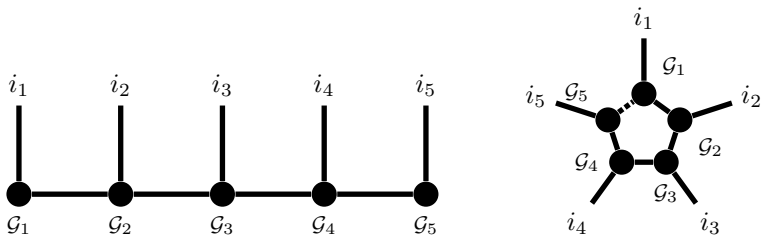
$$\hat{\mathcal{G}}_n[i_n] = A_{n-1}^{-1} \mathcal{G}_n[i_n] A_n, \tag{4.14}$$

without affecting the underlying tensor, as the A_n cancel. This gives us some degrees of freedom in picking cores with desirable properties. One such set of cores is the set of left-canonical cores, with the property that

$$\sum_{i_n} \mathcal{G}_1[i_n]^\top \mathcal{G}_1[i_n] = \boldsymbol{I}. \tag{4.15}$$

This is particularly convenient in the context of normalization for the MPS as applying this identity iteratively to equation (4.11) demonstrates that $\boldsymbol{L}_n = \boldsymbol{I}$ for all n. Similarly, the set of right canonical cores guarantee that $\boldsymbol{R}_n = \boldsymbol{I}$ (Schollwöck 2011). As a consequence, both of these sets of cores lead to automatically normalized MPS tensors. Any tensor train can be restated using canonical cores, up to an external scaling coefficient. In fact, the analytical algorithms automatically calculates a tensor train of this form. The left-canonical set of cores is still not entirely unique, though, as we can apply gauge transforms with orthogonal A_n without affecting the identity. An even stronger, and unique, set of cores is the set that we shall refer to as the Vidal representation or $\Gamma\Lambda$-representation (Schollwöck 2011),

$$\mathcal{T}_{\mathcal{I}} = (\Gamma_1[i_1]\Lambda_1) \dots (\Gamma_{N-1}[i_{N-1}]\Lambda_{N-1})(\Gamma_N[i_N]\Lambda_N) \tag{4.16}$$



(a) A tensor train with $N = 5$. Note that the two outer tensors are 2-tensors.

(b) A tensor ring with $N = 5$. The dotted line highlights the key difference from the tensor train.

Figure 4.3: The tensor train and tensor ring.

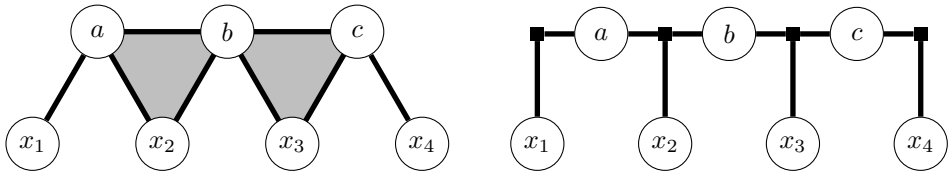
This adds additional parameters to the model, but we constrain Λ_i to be diagonal and impose a double set of constraints on the Γ_n, requiring that

$$\boldsymbol{I} = \sum_{i_n} (\Lambda_{n-1}\Gamma_n[i_n])^\top (\Lambda_{n-1}\Gamma_n[i_n]), \quad \boldsymbol{I} = \sum_{i_n} (\Gamma_n[i_n]\Lambda_n)(\Gamma_n[i_n]\Lambda_n)^\top. \quad (4.17)$$

These conditions relate to canonicity, as collecting the parameters as either $\mathcal{G}_n[i_n] = \Lambda_{n-1}\Gamma_n[i_n]$ or $\mathcal{G}_n[i_n] = \Gamma_n[i_n]\Lambda_n$ yields cores that are either left- or right-canonical, respectively, with respect to the same tensor. Without the $\Gamma\Lambda$-representation, we would have had to go through an expensive conversion procedure. Blending the left- and right-canonical approaches, we can simplify expressions considerably: the marginal becomes simply,

$$p(x_n = i_n) \propto \mathrm{Tr}[\Gamma_n[i_n]^\top \Lambda_{n-1}^2 \Gamma_n[i_n] \Lambda_n^2]. \quad (4.18)$$

The Vidal representation is the gold standard from a computational point of view since reducing the number of consecutive matrix products helps with both the finite-time complexity and numerical issues. In addition to striving for computational benefits, we can also look for parameterizations that induce certain qualitative properties in the tensor. Physicists often use MPS's that encode translation invariance, simply by setting the cores equal to each other for all n (Perez-Garcia et al. 2006), but have also contructed MPS models that are rotationally invariant or otherwise embody physical symmetry properties (Singh, Pfeifer, and Guifré Vidal 2010; Singh, Pfeifer, and Guifre Vidal 2011; Weichselbaum 2012). Attempts to encode symmetry properties of codes and bit-strings has also been attempted (Huckle, Waldherr, and Schulte-Herbrueggen 2013).



(a) Dual graphical model of the tensor train.

(b) The corresponding factor graph.

Figure 4.4: The graphical model corresponding to the tensor train according to the duality theorem.

4.4.2 Tensor Trains as Graphical Models and Efficient Inference

We can apply the duality result from theorem 4.1 to find the dual graphical model of the tensor train (Robeva and Seigal 2017), as in figure 4.4a. To make it easier to parse, we translate the hypergraph to a more relatable factor graph format, depicted in 4.4b (Yedidia, Freeman, and Weiss 2005; Bishop 2006; Wainwright and Jordan 2008). Since each edge of the dual hypergraph corresponds to a clique and a clique potential in this setup, we can just replace each edge by a factor node. We note that in addition to the four random variables x_i that we are modeling, duality introduces three auxiliary variables that we have tentatively labeled a, b, and c. From the factor graph, the new variables appear to transmit dependencies between neighboring random variables. The dependency of x_1 on x_2 and vice versa goes through a, for instance. While the structure is reminiscent of a Hidden Markov Model or other sequential latent model, we note that a HMM of the same size would have another latent variable and almost twice as many factors, but they would all be 2-variable factors. Recall that by tensor arguments, we know that this type of structure is sufficient to model any discrete probability distribution, if only the rank is suitably high. Rank corresponds to the size of the state spaces of a, b, and c in the dual graphical model.

We promised earlier that certain concepts from graphical models would transfer to tensor networks meaningfully. One thing that we know how to do in graphical models is efficient marginalization and exact inference using the junction-tree algorithm (Wainwright and Jordan 2008). Given a tensor network with no dangling edges, we can apply the junction-tree algorithm to get an efficient sequence of marginalization steps. These transfer directly to tensor contractions (Robeva and Seigal 2017). Optimal contraction order is a non-trivial problem in general tensor networks, so getting a good ordering of the contraction steps is valuable, although we are still left with the arduous problem of finding chordal completions as part of running the junction-tree algorithm.

In Robeva and Seigal (2017), they demonstrate this principle directly on the MPS. In short, given a contraction problem similar to the one we solved to compute the normalization constant, they transform it to the dual graphical model and by running the junction tree algorithm they arrive at exactly the same contraction order as we used in equation (4.10) to compute the normalization constant, which has been the standard in the physics community due to its efficiency.

The duality also applies in the opposite direction: (Evenbly and Pfeifer 2013) proves that if every index except those associated with a single tensor can be contracted in time κ, then this can be done in time κ for any other tensor as well, and all of these separate contractions can be computed in 3κ. Whether this result has any application in graphical models is unknown, but by the duality argument it could be transferred.

4.5 Contribution

Although the idea of matrix product states as probabilistic models has started to seep into machine learning (Stoudenmire and Schwab 2016; Han et al. 2017; Pestun and Vlassopoulos 2017), it is still a model with much of its literature firmly rooted in quantum mechanics. Our article *Matrix Product states for inference in discrete probabilistic models* attempts to bridge this divide by offering a solid introduction to the topic for the machine learning community. It is also the first paper to consider variational inference with an MPS as the variational approximation, offering algorithms for calculating unbiased gradients and performing inference.

We also consider differentiable representations for canonical cores, and demonstrate a core design that is simultaneously canonical and relabeling invariant. We discuss the general principles behind cores observing various group symmetries, relating it to the representation theory of groups and the existing theory on continuous symmetries developed in the physics literature (Huckle, Waldherr, and Schulte-Herbrueggen 2013; Weichselbaum 2012; Bridgeman and Chubb 2017).

Chapter 5

Conclusion

The previous three chapters have covered the topics central to our papers and thesis, namely variational inference, probabilistic kernel methods and Gaussian processes, and tensor networks and matrix product states. How the topics relate to the three papers affiliated with this PhD has been described in the individual contribution sections of the chapters, as well as in the thesis overview in the introduction.

While the relationship between the three papers can seem a bit tangential at times, all of our papers deal with models that exhibit strong mutual dependence in the posterior, preventing factorization and each paper tries to deal with this in a novel way. The first paper *Difference-of-convex optimization for variational KL-corrected inference in Dirichlet process mixtures* applies standard mean-field, but with a correction to the bound that like proper marginalization tries to disregard nuisance parameters in an approximately optimal manner. As such we try to bias the optimization routine away from approximations that put undue weight on the nuisance parameters and disregard the dependencies in a principled manner. This results in a fixed point scheme, which unfortunately falls short of expectations by performing identically to the classical coordinate ascent procedure.

This contrasts with the third paper, *Matrix Product states for inference in discrete probabilistic models*, which tries to tackle the variational inference problem

on discrete probabilistic models directly. Discrete models face all of the problems common to structured models, but also very salient issues of combinatorial explosion in the number of parameters needed to describe arbitrary fully-structured systems. To manage this complexity, we need a parametric model which can describe dependencies without compromising on tractability. We find that tensor networks, and matrix product states in particular, could be a viable approach in this regard, by reframing the discrete distributions as tensors and by using tensor rank as a structural constraint to control the computational and structural complexity of the approximation.

in our second paper *Joint expectation kernels* we consider a problem in direct contrast to the discrete models, involving Gaussian processes—elegant continuous non-parametric models often used as probability measures over random functions. They have closed-form inference, but it often comes at a significant computational cost due to the highly dependent random variables it models. We consider how this interfaces with other stochastic variables in a hierarchy, preventing the direct application of the analytical inference formula. By reframing the problem completely, we find an analogous problem where we can apply the analytical formulas, but at the cost of embedding the stochastic components of the lower-level elements in the hierarchy directly into the Gaussian process in a non-probabilistic manner.

The solutions offered by our papers have met with varying degrees of success, but we believe that they have jointly helped explore the space of approximate methods available to researchers. We further believe that all of the papers have laid the groundwork for potentially fruitful future research: the difference-of-convex observation might be exploited more fully by future innovations in the optimization literature, the kernel embedding has laid the groundwork for further exploration of embedded random variables and its relationship to inference and contrasts with distribution-oriented embeddings, and finally our article on matrix product states will hopefully serve as an introduction to the machine learning community, acting as a gateway to the literature and its relationships with probabilistic modeling, and we believe that the class of models could be a very welcome addition to any practitioner's arsenal of models.

Appendix A

Papers

A.1 Difference-of-convex optimization for variational KL-corrected inference in Dirichlet process mixtures

The following paper was published as,

> R Bonnevie, M N Schmidt, and M Mørup (2017). "Difference-of-Convex optimization for variational kl-corrected inference in dirichlet process mixtures". In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6

It is related to background material covered in chapter 2.

2017 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 25–28, 2017, TOKYO, JAPAN

DIFFERENCE-OF-CONVEX OPTIMIZATION FOR VARIATIONAL KL-CORRECTED INFERENCE IN DIRICHLET PROCESS MIXTURES

Rasmus Bonnevie, Morten Mørup, Mikkel N. Schmidt

Technical University of Denmark
Department of Applied Mathematics and Computer Science

ABSTRACT

Variational methods for approximate inference in Bayesian models optimise a lower bound on the marginal likelihood, but the optimization problem often suffers from being non-convex and high-dimensional. This can be alleviated by working in a collapsed domain where a part of the parameter space is marginalized. We consider the KL-corrected collapsed variational bound and apply it to Dirichlet process mixture models, allowing us to reduce the optimization space considerably. We find that the variational bound exhibits consistent and exploitable structure, allowing the application of difference-of-convex optimization algorithms. We show how this yields an interpretable fixed-point update algorithm in the collapsed setting for the Dirichlet process mixture model. We connect this update formula to classical coordinate ascent updates, illustrating that the proposed improvement surprisingly reduces to the traditional scheme.

Index Terms— difference-of-convex optimization, variational inference, collapsed methods, bayesian nonparametrics

1. INTRODUCTION

Although variational inference has been around for a while [1], there has been a surge in interest lately, moving variational inference beyond the traditional mean-field approximation and coordinate-ascent optimization. Recent advances include algorithms for non-conjugate black box inference [2], stochastic optimization in the large data setting [3], and universally applicable probabilistic programming software [4], making inference tractable for complex models such as Bayesian neural networks [5].

Despite these advances, the variational approach hinges on solving a potentially massive, non-convex, and high-dimensional optimization problem. Reducing the parameter space by analytically marginalizing parts of the variational approximation can lead to a more well-behaved objective function, faster convergence, and better solutions [6]. To this end, we adopt the KL-corrected (KLC) bound as our variational objective. It was originally invented for Gaussian processes alone [7], but was later extended to a larger class of conjugate exponential models [8] where it was demonstrated to reduce the optimization space in a principled manner without affecting the set of solutions. Furthermore, it has already been shown to lead to more efficient optimization [8].

Our primary contribution is the realization that the KLC bound has consistent structure when applied to a Dirichlet process mixture, as it decomposes nicely into convex and concave terms. This leads us to consider difference-of-convex (DC) optimization as exemplified in the convex-concave procedure [9] and its generalization to non-differentiable objectives, the aptly named Difference-of-Convex Algorithm (DCA) [10]. We show that this leads to a nice fixed-point mapping which can be expressed as the softmax of a gradient related to the joint distribution.

While superficially different, and derived by a different route, this fixed-point formula turns out to reduce to the classical mean-field update. We investigate under which conditions this holds and find that it is symptomatic of models with exponential family conditionals. We consider the perspectives of this alternate derivation, including how results about convergence can potentially be carried over.

2. THE KL-CORRECTED VARIATIONAL LOWER BOUND

Consider the general Bayesian problem of inferring a distribution over latent variables $\boldsymbol{Z}$ and internal (nuisance) parameters $\boldsymbol{U}$ given observations of a random variable $\boldsymbol{X}$. We can compute the posterior $p(\boldsymbol{Z}, \boldsymbol{U}|\boldsymbol{X})$ up to a constant, but the normalization constant is typically intractable. Variational inference gets around this issue by defining a family of approximations $q(\boldsymbol{Z}, \boldsymbol{U})$ and then minimizing the KL divergence $\mathrm{KL}(q\|p)$. The KL divergence is similarly intractable, but it shares its critical points with the standard variational lower bound:

$$\mathcal{L}_{MF} = \mathbb{E}_q[\ln p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{U})] - \mathbb{E}_q[\ln q(\boldsymbol{Z}, \boldsymbol{U})]\,. \quad (1)$$

Minimizing this non-convex objective with respect to the parameters of q leads to a locally optimal approximation of $p(\boldsymbol{Z}, \boldsymbol{U}|\boldsymbol{X})$. Equation (1) is referred to as a lower bound as it lower bounds the log-evidence $\ln p(\boldsymbol{X})$.

978-1-5090-6341-3/17/$31.00

Suppose we now try to marginalize $\boldsymbol{U}$ prior to doing inference, then the resulting bound has the form

$$\mathcal{L}_C = \mathbb{E}_q\left[\ln \int p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{U})\, \mathrm{d}\boldsymbol{U}\right] - \mathbb{E}_q[\ln q(\boldsymbol{Z})]\,, \quad (2)$$

which unfortunately requires the computation of the expectation of a log-integral. Even if the integral is tractable, the expectation over q often will not be. In the particular case of conjugate exponential family models the integral leads to a compound distribution outside of the exponential family, which means that we lose many of the tractability benefits of working with exponential family models.

This brings us to the KL-corrected bound. The derivation of the KL-corrected lower bound is a form of pseudo-marginalization which reduces the parameter space and leaves a more well-behaved (and still tractable) objective function, but where the inference is still effectively over the original unmarginalized model. There are several ways to derive it, and we will follow Hensman et al. by deriving it by way of an auxiliary bound [8].

The auxiliary bound is derived as a standard lower bound, but for the model conditioned on $\boldsymbol{U}$, instead of on the full joint distribution.

$$\mathcal{L}_1(\boldsymbol{U}) = \mathbb{E}_{q(\boldsymbol{Z})}[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{U}) - \ln q(\boldsymbol{Z})]\,. \quad (3)$$

Note that the bound is a function of $\boldsymbol{U}$. The conditional bound can be transformed into the KL-corrected bound as follows

$$\mathcal{L}_{KL} = \ln \mathbb{E}_{p(\boldsymbol{U})}[\exp(\mathcal{L}_1(\boldsymbol{U}))]\,. \quad (4)$$

Since $\mathcal{L}_1(\boldsymbol{U})$ is a bound on $\ln p(\boldsymbol{X}|\boldsymbol{U})$, the operations above result in $\mathcal{L}_{KL}$ being a bound on the marginal likelihood $\ln p(\boldsymbol{X})$ as desired. The KLC bound is related to the CVB0 approximation [11] as detailed in the original article [8].

2.1. The KLC Bound for the Dirichlet Process Mixture

As an example, we will consider a particular conjugate exponential family model where the KL-corrected bound is computationally advantageous — namely a Dirichlet process mixture model. KLC bounds for finite mixture models have already been covered [8, supplementary], but we will need the KLC bound later so we provide the derivation here for the non-parametrically extended mixture. We will leave the component distribution arbitrary, under the constraint that it is an exponential family distribution with a density of the form

$$p(\boldsymbol{x}_i|\boldsymbol{\eta}_k) = h(\boldsymbol{x}_i)\exp\big(\boldsymbol{\eta}_k^\top T(\boldsymbol{x}_i) - A(\boldsymbol{\eta}_k)\big) \quad (5)$$

We further model each parameter vector $\boldsymbol{\eta}_k$ as being drawn from a common conjugate prior $\boldsymbol{\eta}_k \sim p(\boldsymbol{\eta}|\gamma, \nu)$. We can combine the above into a mixture model using latent indicators $\boldsymbol{Z}$

$$p(\boldsymbol{X}, \{\boldsymbol{\eta}_k\}_{k=1}^\infty|\boldsymbol{Z}) = \prod_{k=1}^\infty \left[\prod_{i=1}^N p(\boldsymbol{x}_i|\boldsymbol{\eta}_k)^{z_{ik}}\right] p(\boldsymbol{\eta}_k|\gamma, \nu) \quad (6)$$

The final component needed is the prior on $\boldsymbol{Z}$. We will employ a stick-breaking representation of the Dirichlet process, but one could just as well use a Dirichlet-multinomial pair for a finite mixture. The stickbreaking distribution takes the form

$$\beta_k \sim \mathrm{Beta}(1, \alpha), \quad z_{ik}|\beta_1, \ldots, \beta_k \sim \mathrm{Cat}(w_k) \quad (7)$$

with dependency through $w_k = \beta_k \prod_{\ell<k}(1-\beta_\ell)$. While this prior has an unbounded number of variables, we will control for this later using the variational approximation so that the bounds only ever have a finite number of terms. Writing out the prior density gives us

$$p(\boldsymbol{Z}|\boldsymbol{\beta}) = \prod_{i=1}^N \prod_{k=1}^\infty \left[\beta_k \prod_{\ell<k}(1-\beta_\ell)\right]^{z_{ik}} = \prod_{k=1}^\infty \beta_k^{m_k}(1-\beta_k)^{m_{k+1}^\infty}, \quad (8)$$

where $m_k = \sum_{i=1}^N z_{ik}$ and $m_{k+1}^\infty = \sum_{\ell=k+1}^\infty m_\ell$.

Collecting $\boldsymbol{U} = (\{\boldsymbol{\eta}_k\}_{k=1}^\infty, \boldsymbol{\beta})$, we can compute the $\mathcal{L}_1(\boldsymbol{U})$ conditional bound, under an exponential family variational approximation $q(\boldsymbol{Z}|\boldsymbol{\mu})$ parametrized by mean parameters $\boldsymbol{\mu}$, resulting in

$$\mathcal{L}_1(\boldsymbol{U}) = C + \sum_{k=1}^\infty \left[\boldsymbol{\eta}_k^\top \bar{T}_k - \bar{m}_k A(\boldsymbol{\eta}_k)\right] + \sum_{k=1}^\infty \left[\bar{m}_k \ln(\beta_k) + \bar{m}_{k+1}^\infty \ln(1-\beta_k)\right] + \mathcal{H}_q(\boldsymbol{\mu}), \quad (9)$$

where $C = \sum_{i=1}^N \ln h(\boldsymbol{x}_i)$, $\bar{T}_k = \sum_{i=1}^N \mathbb{E}_q[z_{ik}]\, T(\boldsymbol{x}_i)$, $\bar{m}_k = \sum_{i=1}^N \mathbb{E}_q[z_{ik}]$, and $\bar{m}_{k+1}^\infty = \sum_{\ell=k+1}^\infty \bar{m}_\ell$.

To compute the KL-corrected bound we will split up the $\mathcal{L}_1$ bound into terms depending on $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$. Taking the exponential as in equation (4) gives us expressions with the same functional forms as the original distributions, allowing us to integrate over the appropriate conjugate priors, yielding factors

$$\prod_{k=1}^\infty \frac{e^{A_0(\boldsymbol{\gamma}+\bar{T}_k, \nu+\bar{m}_k)}}{e^{A_0(\boldsymbol{\gamma},\nu)}}, \quad \prod_{k=1}^\infty \frac{B(1+\bar{m}_k, \alpha+\bar{m}_{k+1}^\infty)}{B(1,\alpha)}. \quad (10)$$

for the likelihood and latent distributions, respectively. Here, A_0 is the log-normalizer of the conjugate prior to $\boldsymbol{\eta}$ and B is the beta function, i.e. the normalizer of the Beta distribution..

The KL-corrected bound now follows naturally from the definition.

$$\mathcal{L}_{KL} = \sum_{k=1}^\infty \big[A_0(\boldsymbol{\gamma}+\bar{T}_k, \nu+\bar{m}_k) + \ln B(1+\bar{m}_k, \alpha+\bar{m}_{k+1}^\infty)\big] + \mathcal{H}_q(\boldsymbol{Z}) + \text{const.} \quad (11)$$

2.2. Difference-of-Convex Structure of the KLC Bound

Our key observation is that both A_0 and $\ln B$ are the log-normalizers of exponential family models and are thus known to be convex [12]. Since $\bar{T}_k$ and $\bar{m}_k$ are linear functions in $\mu_{ik} \equiv \mathbb{E}_q[z_{ik}]$, we know that their composition with a convex function results in something that is also convex in μ_{ik} [13]. Since the sum likewise preserves convexity, the whole sum is convex.

Furthermore, if $q(\boldsymbol{Z})$ is an exponential family with mean parametrization $\boldsymbol{\mu}$ and log-normalizer A_q, then it can also be shown that [12, theorem 3.4]

$$-A_q^*(\boldsymbol{\mu}) = \mathcal{H}_q(\boldsymbol{\mu}), \tag{12}$$

where A_q^* denotes the convex conjugate of the log-normalizer. Since the convex conjugate is always convex, we can conclude that the entropy is concave for an exponential family [12].

To summarize, the bound is made up of a convex and a concave part; additional structure we should do our best to exploit. To stay true to the optimization literature, we will consider minimization of $-\mathcal{L}_{KL}$ from here on out, resulting in the following (flipped) decomposition

$$-\mathcal{L}_{KL} = f_{vex} + f_{cave} - C, \quad f_{vex} = -\mathcal{H}_q(\boldsymbol{\mu}), \tag{13}$$

$$f_{cave} = -\sum_{k=1}^{\infty} \left[A_0(\boldsymbol{\gamma} + \bar{T}_k, \nu + \bar{m}_k) + \right. \tag{14}$$

$$\left. \ln B(1 + \bar{m}_k, \alpha + \bar{m}_{k+1}^{\infty}) \right]. \tag{15}$$

3. CONVEX-CONCAVE PROCEDURE

Optimization problems with a mix of convex and concave terms are denoted as difference-of-convex problems (DC). Technically, any non-convex smooth problem is a DC problem as functions can be decomposed into regions of positive and negative curvature, but the decomposition is not always obvious [9, 10].

The **c**onvex-**c**on**c**ave **p**rocedure (CCCP) is a straightforward algorithm for DC problems [9]. The core idea is that a stationary point for a difference function occurs when the gradients of the two terms match, i.e.

$$0 = \nabla \left(f_{vex} + f_{cave} \right) \Leftrightarrow \nabla f_{vex} = -\nabla f_{cave}. \tag{16}$$

The CCCP algorithm simply turns this premise into an implicit fixed-point scheme

$$\nabla f_{vex}(\boldsymbol{\mu}_{t+1}) = -\nabla f_{cave}(\boldsymbol{\mu}_t), \tag{17}$$

so $\boldsymbol{\mu}_{t+1}$ is picked so that the convex gradient matches the negative concave gradient at time t. While this might look arbitrary, this in fact elegantly exploits the features of the function's convex-concave nature, ensuring a monotonously decreasing sequence [9].

An equivalent, but slightly more approachable, interpretation of CCCP is as a sequential optimization problem, where

$$\boldsymbol{\mu}_{t+1} = \operatorname*{argmin}_{\boldsymbol{\mu}} f_{vex}(\boldsymbol{\mu}) + \tilde{f}_{cave}^{(\boldsymbol{\mu}_t)}(\boldsymbol{\mu}). \tag{18}$$

where we have linearized the concave part around $\boldsymbol{\mu}_t$ as $\tilde{f}_{cave}^{(\boldsymbol{\mu}_t)}(\boldsymbol{\mu}) = \big(f_{cave}(\boldsymbol{\mu}_t) + (\boldsymbol{\mu} - \boldsymbol{\mu}_t)^\top \nabla f_{cave}(\boldsymbol{\mu}_t)\big)$. Since the concave part is linearized it becomes trivially convex (in addition to concave), and the complete objective is then unequivocally a convex function, leading to a simplified problem where the full brunt of convex optimization can be brought into play. Since the linearization of a concave function upper bounds the concave function itself this provides an illustration of why the sequence is monotonously decreasing (see figure 1). for more details, see [14, 9, 10].

3.1. Necessary Conditions on the Variational Distribution

So far, we have left the variational approximation $q(\boldsymbol{Z}|\boldsymbol{\mu})$ vague. With the above in place, we see that to apply CCCP to our bound, there are two key restrictions (and an additional facilitator).

Expectations Linear in the Parameters We are relying on the transparent relationship $\boldsymbol{\mu}_{ik} = \mathbb{E}[z_{ik}]$ between parameters and expected latent variables. This could be relaxed a bit — the expectation could be any linear function of the parameters. In fact, it could even be a convex or concave function of the parameters following the standard composition theorems, assuming some further conditions hold [13].

Concave Entropy The variational approximation needs to have concave entropy. The entropy function is concave for exponential family models [12], and entropy in general is concave in the space of distributions, but we have been unable to document that this holds for all distributions outside of the exponential family, as well as all possible parameterizations.

(Tractable Inverse Gradient Map) Ideally, we would also like to know the inverse entropy-gradient map. This turns out to be well-known for many exponential families, but is likely unavailable for many more interesting variational approximations. Fortunately, we will still be able to solve the sequential problem in equation (18) efficiently if q obeys the other conditions, so the inverse map is not strictly necessary.

We can find at least one simple variational approximation obeying the above conditions in the form of the widely used product of single-sample multinomials (i.e. categorical distributions).

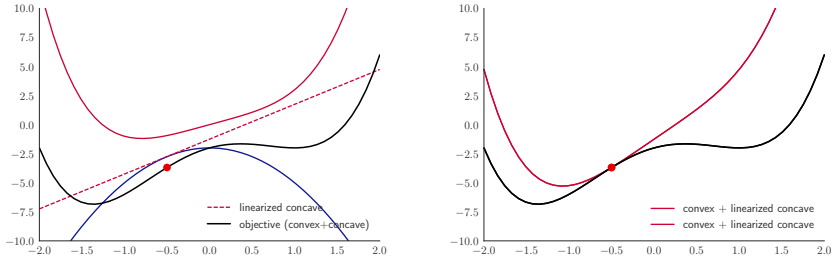


Fig. 1. The sequential interpretation minimizes the objective (black) by constructing an upper bound. The concave term is linearized to yield a convex upper bound (red; right).

3.2. Fixed-point Update for the KLC Bound

To formulate our main result, we rewrite equation (17) following [9],

$$\boldsymbol{\mu}_{t+1} = \left[\nabla f_{vex}\right]^{-1}\left(-\nabla f_{cave}(\boldsymbol{\mu}_t)\right). \tag{19}$$

Since the KLC bound consists of log-normalizers, we just need to be able to compute gradients of exponential family log-normalizers to compute the gradient of the bound. This makes it relevant to mention the following relationships between the (arbitrary exponential family) distribution's log-normalizer A, its natural parameters $\boldsymbol{\eta}$, and its dual parametrization in mean parameters $\boldsymbol{\mu}$ [12]

$$\left[\nabla A\right]^{-1}(\boldsymbol{\mu}) = \nabla A^*(\boldsymbol{\mu}) = \boldsymbol{\eta}, \tag{20}$$

$$\left[\nabla A^*\right]^{-1}(\boldsymbol{\eta}) = \nabla A(\boldsymbol{\eta}) = \boldsymbol{\mu}, \tag{21}$$

illustrating key symmetries found in exponential family models. Recall that $f_{vex} = -\mathcal{H}_q(\boldsymbol{\mu}) = A_q^*(\boldsymbol{\mu})$, so that $\left[\nabla f_{vex}\right]^{-1} = \nabla A_q$ by the above. Then CCCP yields

$$\boldsymbol{\mu}_{t+1} = \nabla A_q(-\nabla f_{cave}(\boldsymbol{\mu}_t)). \tag{22}$$

If we compare this to the second identity in (20), it appears that $-\nabla f_{cave}(\boldsymbol{\mu}_t)$ is in some sense representing a set of natural parameters. At the fixed point $\boldsymbol{\mu}^*$ of the update rule, it must in fact be the exact corresponding natural parameter $\boldsymbol{\eta}^*$, i.e. $-\nabla f_{cave}(\boldsymbol{\mu}^*) = \boldsymbol{\eta}^* = \nabla A_q^*(\boldsymbol{\mu}^*)$.

We can make the update rule a bit more explicit, but first we have to handle the normalization constraints $\sum_{k=1}^K \mu_{ik} = 1$. We add Lagrangian terms to the convex terms such that

$$f_{vex} = \sum_{i=1}^N \sum_{k=1}^K \mu_{ik} \ln \mu_{ik} + \sum_{i=1}^N \lambda_i \left(\sum_{k=1}^K \mu_{ik} - 1\right). \tag{23}$$

Taking the derivative yields

$$g_{ik} = \frac{\partial}{\partial \mu_{ik}} f_{vex} = \ln \mu_{ik} + 1 + \lambda_i \Leftrightarrow \mu_{ik} = \frac{\exp(g_{ik} - 1)}{\exp(\lambda_i)}. \tag{24}$$

Applying the constraint, we get that

$$\mu_{ik} = \frac{\exp(g_{ik})}{\sum_{k=1}^K \exp(g_{ik})}, \tag{25}$$

which is the softmax function $\mathcal{S}(\cdot)$, so we can write the actual CCCP update formula (equation (19)) as

$$\boldsymbol{\mu}_{t+1} = \mathcal{S}(-\nabla f_{cave}(\boldsymbol{\mu}_t)) = \mathcal{S}(\nabla_{\boldsymbol{\mu}} \ln p(\boldsymbol{X}, \mathbb{E}_{\boldsymbol{\mu}}[\boldsymbol{Z}]))|_{\boldsymbol{\mu}=\boldsymbol{\mu}_t}. \tag{26}$$

This is reminiscent of exponentiated gradient algorithms which show up when objectives are regularized with Kullback-Leibler divergences [15, 16]. Since the KL divergence includes an entropy term it makes sense that they appear similar.

4. CONNECTING THE KLC UPDATES WITH MEAN-FIELD COORDINATE ASCENT

The original solution to the mean-field variational inference problem in the uncollapsed setting was to apply coordinate ascent, updating each distribution in turn. This procedure was often tractable for conjugate exponential family models, if sometimes convoluted.

In general, by taking the derivative of the lower bound with respect to the parameters controlling the distribution over model variable $\theta_i \in \{\theta_j\}_{j=1}^N$ and setting the derivative to zero, we can find that the optimal variational approximation is[17]

$$q(\theta_i) \propto \exp(\mathbb{E}_q[\ln p(\theta_i | \mathcal{D}, \{\theta_j\}_{i \neq j}) \mid \theta_i]), \tag{27}$$

where $\mathcal{D}$ is the set of observed variables. Usually, mean field assumptions are exploited to ensure that the expectations are tractable, but if the expectations are computable without that assumption then the parameters can be updated in blocks.

For our mixture, the expectation resolves to

$$\mathbb{E}_q\big[\ln p(z_i|\boldsymbol{X}, \boldsymbol{Z}_{\setminus i}, \boldsymbol{U}) \mid \boldsymbol{Z}\big] = \tag{28}$$

$$\sum_{k=1}^{K} z_{ik} \left(T(\boldsymbol{x}_i)^\top \mathbb{E}[\boldsymbol{\eta}_k] - \mathbb{E}[A(\boldsymbol{\eta}_k)] + \mathbb{E}[\ln w_k(\boldsymbol{\beta})]\right) \tag{29}$$

where we will denote the term in the parenthesis by $\ln \tilde{\pi}_{ik}$, which is understood to be the log of the unnormalized probability parametrizing the multinomial variational approximation over z_i. If we take the softmax of $\ln \tilde{\pi}_{ik}$ we recover the distribution itself. Let us compare this to $-\nabla f_{cave}$. We will consider its terms individually, starting with the log-normalizer terms involving A_0. Taking the gradient, we get

$$\frac{\partial}{\partial \mu_{ik}} A_0(\boldsymbol{\gamma} + \bar{T}_k, \nu + \bar{m}_k) = \tag{30}$$

$$\nabla_{\boldsymbol{\gamma},\nu} A_0^\top \begin{pmatrix} T(\boldsymbol{x}_i) \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbb{E}[\boldsymbol{\eta}_k] \\ -\mathbb{E}[A(\boldsymbol{\eta}_k)] \end{pmatrix}^\top \begin{pmatrix} T(\boldsymbol{x}_i) \\ 1 \end{pmatrix}. \tag{31}$$

These terms exactly match the ones found in the coordinate ascent update.

Following a similar process, if we take the derivatives of the log-beta terms — being log-normalizers of beta distributions — in f_{cave}, we recover the latent terms in the ascent updates

$$\frac{\partial}{\partial \mu_{i\ell}} \ln B(1 + \bar{m}_k, \alpha + \bar{m}_{k+1}^{\infty}) = \begin{pmatrix} \mathbb{E}[\ln \beta_k] \\ \mathbb{E}[\ln(1-\beta_k)] \end{pmatrix}^\top \begin{pmatrix} 1[\ell = k] \\ 1[\ell < k] \end{pmatrix}, \tag{32}$$

as the latent terms in the ascent formula can be expanded as

$$\mathbb{E}[\ln w_k(\boldsymbol{\beta})] = \mathbb{E}[\ln \beta_k] + \sum_{\ell<k} \mathbb{E}[\ln(1-\beta_k)]. \tag{33}$$

Thus we can conclude that the CCCP strategy exactly matches classical coordinate ascent. This conclusion hinges on the expectations being over the same variational distribution $q(\boldsymbol{U})$, but the KL-corrected bound implicitly always uses the optimal approximation $q^*(\boldsymbol{U})$ so if we take coordinate ascent steps to maximize $q(\boldsymbol{U})$ before updating $\boldsymbol{\mu}$, then the expectations will always match.

To investigate this further, recall that in the uncollapsed setting we need to find both a variational approximation $q(\boldsymbol{U})$ over the component parameters, as well as a distribution over the clusters parameterized by $\boldsymbol{\mu}$. To every state $\boldsymbol{\mu}_t$, there is an optimal setting of the variational approximation $q(\boldsymbol{U})$; we use $\Lambda(\boldsymbol{\mu}_t)$ to denote the implicit map that maps $\boldsymbol{\mu}_t$ to the optimal $q^*(\boldsymbol{U})$. Let us use that the entropy $(-f_{vex})$ is a term $\mathcal{L}_{MF}$ and $\mathcal{L}_{KL}$ have in common and define a decomposition $-\mathcal{L}_{MF}(\boldsymbol{\mu}, \Lambda(\boldsymbol{\mu}_t)) = -\mathcal{E}(\boldsymbol{\mu}, \Lambda(\boldsymbol{\mu}_t)) + f_{vex}(\boldsymbol{\mu})$ where $\mathcal{E}$ is the average energy — the first argument $\boldsymbol{\mu}$ is identified with a distribution over $\boldsymbol{Z}$, while the second argument is the distribution over $\boldsymbol{U}$ which we set to the optimal value with respect to a previous iterate $\boldsymbol{\mu}_t$, using the implicit map $\Lambda(\cdot)$. If $\boldsymbol{\mu}_*$ is the optimum of the bound, the first-order optimality condition for $\mathcal{L}_{MF}$ with respect to $\boldsymbol{\mu}$ states that

$$\nabla \mathcal{E}(\boldsymbol{\mu}_*, \Lambda(\boldsymbol{\mu}_t)) = \nabla f_{vex}(\boldsymbol{\mu}_*) \tag{34}$$

Now recall that CCCP can be interpreted as a sequential convex problem (equation (18)) with a linearized concave component $\tilde{f}_{cave}(\boldsymbol{\mu})$. We then have the exact same optimality condition, but at a potentially different point: $-\nabla \tilde{f}_{cave}(\tilde{\boldsymbol{\mu}}_*) = \nabla f_{vex}(\tilde{\boldsymbol{\mu}}_*)$. Furthermore, recall that the two bounds have matching values and gradients at $\boldsymbol{\mu}_t$ by construction, i.e. $\nabla \mathcal{L}_{KL}(\boldsymbol{\mu}_t) = \nabla \mathcal{L}_{MF}(\boldsymbol{\mu}_t)$, which means the energy must match the linearized concave component

$$-\nabla \tilde{f}_{cave}(\tilde{\boldsymbol{\mu}}_*) = \mathcal{E}(\boldsymbol{\mu}_t, \Lambda(\boldsymbol{\mu}_t))$$

since $\tilde{f}_{cave}$ is linear, its gradient is constant, so if $\tilde{\boldsymbol{\mu}}_* = \boldsymbol{\mu}_*$, we have the peculiar property that $\nabla \mathcal{E}(\boldsymbol{\mu}_t, \Lambda(\boldsymbol{\mu}_t)) = \nabla \mathcal{E}(\boldsymbol{\mu}_*, \Lambda(\boldsymbol{\mu}_t))$, i.e. the gradient of the energy is constant as well. This hints at "hidden linearity" in the average energy.

A partial explanation comes from considering the case where the distribution $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{U})$ is in an exponential family together with its prior $p(\boldsymbol{Z}|\nu)$. Following Hoffman et al. [3], the *natural* gradient $\tilde{\nabla}$ of the lower bound for an exponential family with parameters $\boldsymbol{\eta}$ is

$$\tilde{\nabla}_{\boldsymbol{\eta}} \mathcal{L}_{MF}(\boldsymbol{\eta}) = \boldsymbol{\eta} - \mathbb{E}_{q(\nu)}[\nu] = \nabla_{\boldsymbol{\mu}} \mathcal{L}_{MF}(\boldsymbol{\eta}(\boldsymbol{\mu})), \tag{35}$$

where the last equality can be proven using the chain rule [8]. Since the $\boldsymbol{\eta}$ term comes from the entropy, the natural gradient in the energy does indeed appear to be constant. So from this it's clear that mean parameter gradients of the average energy are constant when the conditionals are exponential family distributions.

We note that this identity between the two algorithms has its benefits and can provide new angles of attack for theoretical problems concerning variational inference. As an example, convergence for CCCP and other bound optimization algorithms was investigated by Salakhutdinov et al. [14]. Finally, we should mention that this is not the first connection found between the coordinate ascent updates and other optimization paradigms. Sato deduced that the coordinate ascent updates were similarly identical to natural gradient steps with stepsize 1 [18]. By transitivity our iteration formula is then also identical to a unit natural gradient step.

5. CONCLUSION

The main result of this paper is the demonstration that the KL-corrected bound for the Dirichlet process mixture inher-

its structure from the original variational problem and can be partitioned into convex and concave parts.

We argue that additional information available about an objective function should be exploited to the extent possible, and the difference-of-convex literature indicates that the above split can lead to improved non-convex optimization.

Applying the CCCP algorithm leads to a general analytical fixed-point update formula. The update formula is shown to match standard variational Bayes updates, and thus provides a new angle of attack on the variational problem, which can potentially be extended to models beyond the classical mixture model.

To truly surpass the existing inference schemes it appears that we need difference-of-convex algorithms that can take advantage of second-order derivatives. Unfortunately, to the best of our knowledge, the DC optimization literature has yet to find algorithms improving on CCCP/DCA. We hope that future research will either uncover new ways to exploit the difference-of-convex structure, or that the connections with DC optimization can provide a new fruitful avenue for the analysis of collapsed variational Bayes.

6. REFERENCES

[1] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1 Nov. 1999.

[2] Rajesh Ranganath, Sean Gerrish, and David M Blei, "Black box variational inference," 31 Dec. 2013.

[3] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley, "Stochastic variational inference," *Journal of machine learning research: JMLR*, vol. 14, no. 1, pp. 1303–1347, May 2013.

[4] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei, "Automatic differentiation variational inference," 2 Mar. 2016.

[5] Diederik P Kingma and Max Welling, "Auto-Encoding variational bayes," 20 Dec. 2013.

[6] Yee W Teh, David Newman, and Max Welling, "A collapsed variational bayesian inference algorithm for latent dirichlet allocation," in *Advances in Neural Information Processing Systems 19*, B Schölkopf, J C Platt, and T Hoffman, Eds., pp. 1353–1360. MIT Press, 2007.

[7] Nathaniel J King and Neil D Lawrence, "Fast variational inference for gaussian process models through KL-Correction," in *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, pp. 270–281. Springer Berlin Heidelberg, 1 Jan. 2006.

[8] James Hensman, Magnus Rattray, and Neil D Lawrence, "Fast variational inference in the conjugate exponential family," in *Advances in Neural Information Processing Systems 25*, F Pereira, C J C Burges, L Bottou, and K Q Weinberger, Eds., pp. 2888–2896. Curran Associates, Inc., 2012.

[9] A L Yuille and Anand Rangarajan, "The concave-convex procedure," *Neural computation*, vol. 15, no. 4, pp. 915–936, Apr. 2003.

[10] Le Thi Hoai An and Pham Dinh Tao, "The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems," *Annals of Operations Research*, vol. 133, no. 1-4, pp. 23–46, 2005.

[11] Katsuhiko Ishiguro, Issei Sato, and Naonori Ueda, "Averaged collapsed variational bayes inference," *Journal of machine learning research: JMLR*, vol. 18, no. 1, pp. 1–29, 2017.

[12] Martin J Wainwright and Michael I Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1-2, pp. 1–305, Jan. 2008.

[13] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge Univ. Pr, 2004.

[14] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani, "On the convergence of bound optimization algorithms," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 2003, UAI'03, pp. 509–516, Morgan Kaufmann Publishers Inc.

[15] David P Helmbold, Robert E Schapire, Yoram Singer, and Manfred K Warmuth, "A comparison of new and old algorithms for a mixture estimation problem," *Machine learning*, vol. 27, no. 1, pp. 97–119, 1997.

[16] Amir Globerson, Terry Y Koo, Xavier Carreras, and Michael Collins, "Exponentiated gradient algorithms for log-linear structured prediction," in *Proceedings of the 24th International Conference on Machine Learning*, New York, NY, USA, 2007, ICML '07, pp. 305–312, ACM.

[17] Christopher M Bishop, *Pattern Recognition and Machine Learning*, Springer, 17 Aug. 2006.

[18] Masa-Aki Sato, "Online model selection based on the variational bayes," *Neural computation*, vol. 13, no. 7, pp. 1649–1681, 2001.

A.2 Joint expectation kernels

The following paper is yet to be published. It has been submitted to NIPS 2018. It is related to background material covered in chapter 3.

Joint Expectation Kernels

Rasmus Bonnevie
DTU Compute, Cognitive Systems
Technical University of Denmark
Copenhagen, Denmark
rabo@dtu.dk

Mikkel N. Schmidt
DTU Compute, Cognitive Systems
Technical University of Denmark
Copenhagen, Denmark
mnsc@dtu.dk

Abstract

A kernel mean embedding (KME) represents a probability distribution as a point in a reproducing kernel Hilbert space, and naturally induces a kernel on the distributions. We propose a new joint expectation kernel which is defined over random variables rather than probability distributions. We argue that our proposed joint expectation kernel, in contrast to the KME, is better suited for problems such as regression over uncertain inputs. We show how the proposed kernel can handle censored and uncertain data, and we highlight how the KME converges to a zero process in the high noise regime and artificially inflates observations in the low-data regime as two qualitative flaws.

1 Introduction

With kernel methods, many classical learning problems have been reframed in terms of datum-to-datum similarities—or inner products, to be precise—rather than by an explicit feature representation. Using an appropriate kernel, algorithms can be applied directly to new classes of non-vectorial datasets, such as graphs [Kondor and Lafferty, 2002, Kondor and Pan, 2016], strings [Collins and Duffy, 2001, Eskin et al., 2003], and probability distributions [Muandet et al., 2012, Flaxman et al., 2015, Muandet et al., 2016]. In this paper we explore the uses of probability space kernels, that enable us to handle data with inherent uncertainty without sacrificing tractability. Compared to a pure probabilistic model where uncertain input points must be sampled or approximated as part of the inference process, the kernel based methods operate directly on noisy or 'smeared out' data points. Applications range from classical regression and classification with input uncertainty, over Bayesian (Gaussian process) smoothed regression, to regression on the output of latent quantities e.g. represented by MCMC samples.

Whereas standard probability space kernels such as kernel mean embeddings [Smola et al., 2007] implicitly take expectations prior to forming the inner product, our proposed kernel averages over the inner products instead. This design allows our joint expectation kernel to take covariance into account, which KME completely ignores. Ignoring covariance can be problematic for example when we have two random variables with identical marginals: KME will conflate the two as it identifies them by the marginal distribution, while our proposal can distinguish between the points as $\mathrm{Cov}(X, Y)$ is different from $\mathrm{Cov}(X, X)$. As such, our proposed kernel is well-suited for tasks where we observe random variables, such as if we have uncertain or censored data.

In the Gaussian setting the joint expectation kernel is particularly applicable, and the explicit incorporation of uncertainty allows us to build models that take the output of jointly Gaussian generative models as input. In particular, we develop a novel probabilistic kernel model where the output of an arbitrary Gaussian process is fed as input to another Gaussian process using the proposed kernel. This yields a novel and flexible family of kernel models which we call nested Gaussian processes.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Probability Space Kernels

By probability space kernels we generally understand kernels that take a random variable $X \sim p_X(X)$ or its probability distribution $p_X(\cdot)$ as input. The most prominent framework for probability space kernels is kernel mean embeddings (KME) [Smola et al., 2007], where a random variable X is characterized by its embedding μ_X,

$$\mu_X = \mathbb{E}_X[k(X, \cdot)]\,. \tag{1}$$

Here $k(\cdot, \cdot)$ is a kernel defined on the same space as X, which we will refer to as the embedding kernel. As μ_X is an element of an RKHS $\mathcal{H}$, it has a natural inner product,

$$k_{\mathrm{KME}}(X, Y) = \langle \mu_X, \mu_Y \rangle_{\mathcal{H}} = \mathbb{E}_Y[\mathbb{E}_X[k(X, Y)]]\,. \tag{2}$$

The kernel mean embedding provides a natural way to handle distributions in an RKHS setting and there is a lot of supporting theory. For a class of kernel functions called characteristic kernels, the mean embedding completely defines the probability distribution, so that $\|\mu_X - \mu_Y\|_{\mathcal{H}} = 0$ implies that X and Y are identically distributed [Sriperumbudur et al., 2011, Muandet et al., 2016].

Another perspective comes from Gaussian processes and Bayesian quadrature, where it is known that if Gaussian process $f(x)$ is generated from kernel k, then the kernel between linear transformations $\mathcal{L}$ of the process $[\mathcal{L}_x f](x)$ and $[\mathcal{L}_y f](y)$ is $\langle \mathcal{L}_x k(x, \cdot), \mathcal{L}_y k(y, \cdot) \rangle_{\mathcal{H}}$ Briol et al. [2015]. As expectation $\mathbb{E}[\cdot]$ is a linear operator, we see that $\langle \mu_X, \mu_Y \rangle$ is the induced covariance between $\mathbb{E}_X[f(X)]$ and $\mathbb{E}_Y[f(Y)]$. So the inner product on embedded distributions is derived from the inner product on expectations over test functions. This is the same line of thought behind characterizing random variables in terms of moments and characteristic functions, which is analyzed more deeply in the review on KMEs by Muandet et al. 2016.

We have highlighted a technical detail in the above kernel definition by writing the expectation as two marginal expectations: A KME kernel generally does not take mutual covariance into account as it is fundamentally a measure of similarity between marginal distributions. To be clear, there do exist kernel mean embeddings of joint distributions as well, see e.g. [Muandet et al., 2016], but this is a separate issue from that of defining a kernel between individual covariant random variables.

2.1 Joint Expectation Kernel

The novel joint expectation kernel (E-kernel) is straightforwardly defined as

$$k_{\mathrm{E}}(X, Y) = \mathbb{E}_{X,Y}[k(X, Y)]\,, \tag{3}$$

where the expectation is now over the joint distribution of X and Y and $k(\cdot, \cdot)$ is an arbitrary embedding kernel as for KME. X and Y are here arbitrary random variables. As opposed to the KME kernel, we take the expectations of the inner product, rather than the inner product between expectations.

The kernel's positive semidefiniteness (PSD) naturally follows from that of the embedding kernel. If $\boldsymbol{K}_{\mathrm{E}} = \mathbb{E}[\boldsymbol{K}]$ is taken to be a $N \times N$ matrix populated by $[\boldsymbol{K}_{\mathrm{E}}]_{ij} = \mathbb{E}[k(X_i, X_j)]$, and X_i are random variables on the same domain, then

$$\boldsymbol{v}^\top \mathbb{E}[\boldsymbol{K}]\boldsymbol{v} = \mathbb{E}\big[\boldsymbol{v}^\top \boldsymbol{K}\boldsymbol{v}\big] \geq 0, \forall \boldsymbol{v} \in \mathbb{R}^D, \tag{4}$$

with the final inequality following as the expectation is over a non-negative random variable, as $\boldsymbol{v}^\top \boldsymbol{K}\boldsymbol{v} \geq 0$ by virtue of the embedding kernel being PSD.

A constructive, but less explicit, approach would be to recall that random variables X, Y are really functions $X(\omega), Y(\omega)$ from a sample space $\omega \in \Omega$ to a measurable space such as the reals [Casella and Berger, 2002]. A kernel related to our E-kernel could then follow from using the standard kernel construction rule $k(x, y) = k_0(f(x), f(y))$ [Bishop, 2006], with x and y being the random variables, and $f(\cdot) = g_\omega(\cdot)$ being the evaluation functional at ω. This would be PSD for fixed ω, and we could then form the E-kernel by integrating over Ω. So the kernel is intuitively an aggregate measure of "event similarity".

2.1.1 Contrasts with KME

The E-kernel is a qualitatively different kernel from the KME, with each kernel being appropriate for different tasks. Even in the minimal setting where all of the variables are independent, the E-kernel does not reduce to the KME. In the case of zero covariance, the two kernels are indeed equal for off-diagonal elements where the zero covariance conditions hold sway, but on the diagonal each variable will trivially be perfectly correlated with itself so that for a stationary embedding kernel $k(x, y) = k(x - y)$,

$$\underbrace{k(0)}_{k_{\mathrm{E}}(X,X)} \neq \underbrace{\mathbb{E}_X[\mathbb{E}_{X'}[k(X, X')]]}_{k_{\mathrm{KME}}(X,X)}, \tag{5}$$

where $X' \overset{(D)}{=} X$ is an "independent copy" of X with the same distribution.

A few key differences between KME and E-kernel:

Limiting behavior As the input variance increases to infinity, the probability that distance $\Delta = X - Y$ is large will also grow. For a kernel function that goes to 0 as the distance increases, this implies that the kernel between two independent random variables will vanish in the limit. The kernel matrix for the KME kernel will converge to a 0 matrix and thus a zero process. The E-kernel on the other hand converges to a scaled identity matrix with $k(0)$ on the diagonal, corresponding to a Gaussian noise process. So if we pick a random point from the domain, the KME kernel will become increasingly certain that it has value 0 a priori.

Inflation effect If we split the kernel as $k(x, y) = \langle k(x, \cdot), k(y, \cdot)\rangle$ and take μ_X to be the kernel mean embedding from equation (1), then we can consider the difference in behavior for a very simple predictive task: given one observation at stochastic unknown variable X of value $y = f(X)$, then looking at the posterior predictive mean at $x^* = 0$ of a Gaussian process trained with these two kernels yields:

$$\hat{y}_{\mathrm{E}} = \frac{\langle k(0, \cdot), \mu_X\rangle}{\langle k(0, \cdot), k(0, \cdot)\rangle} y \tag{6}$$

$$\hat{y}_{\mathrm{KME}} = \frac{\langle k(0, \cdot), \mu_X\rangle}{\langle \mu_X, \mu_X\rangle} y = \frac{\|k(0, \cdot)\|^2}{\|\mu_X\|^2} \hat{y}_{\mathrm{E}}. \tag{7}$$

Here $\hat{y}_{\mathrm{E}}$ is relatively stable with respect to μ, as the prediction only scales with the inner product which decreases with increasing variance of X (as variance 0 maximizes the inner product). $\hat{y}_{\mathrm{KME}}$ scales this prediction with the inverse of the squared norm of μ. If the norm of μ decreases with increasing variance, this inflates the prediction, possibly increasing the prediction beyond the originally observed quantity. Applying Cauchy-Schwartz to the coefficient of $\hat{y}_{\mathrm{E}}$ we have

$$\left| \frac{\langle k(0, \cdot), \mu_X\rangle}{\langle k(0, \cdot), k(0, \cdot)\rangle} \right| \leq \frac{\|\mu_X\|}{\|k(0, \cdot)\|}, \tag{8}$$

which demonstrates that even in the extreme case, the prediction will shrink towards the prior as input variance increases. As $\hat{y}_{\mathrm{KME}}$ inverts the ratio of this bound, there is theoretically no bound to the inflation effect of the KME prediction, although in practice it will be controlled by the choice of kernel.

For Gaussian process regression with an E-kernel, observing a point with extremely high variance means that the point could be anywhere. Correlation with other points vanish, but we can still expect the value of the point to be distributed like the prior marginal distribution. Based on the first difference mentioned above, this argues in favor of the E-kernel. Likewise, if we have no idea about where the point is, our mean belief should fall back towards the prior. This conflicts with the inflation effect, once again arguing in favor of the E-kernel.

Finally, it should be noted that the E-kernels can do things that the KME simply cannot. In particular, we will make the case that SEEK is appropriate for regression on spaces of random variables and points with uncertain location, while KME is appropriate for regression on integrals and distributions.

2.2 Gaussian case: The SEEK kernel

As a useful case study, take $\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ to be jointly Gaussian random variables taking values $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^D$ and take the embedding kernel to be a squared exponential kernel k_{SE}. These assumptions jointly define the Squared Exponential E-kernel (SEEK) which we will derive below. As a convenience, we will write the squared exponential kernel as a scaled Gaussian, in what is sometimes referred to as the expected kernel trick [Turner, 2012]

$$k_{SE}(\boldsymbol{x}, \boldsymbol{y}) = \lambda \mathcal{N}(\boldsymbol{0}; \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{\Sigma}_0) = \tag{9}$$

$$\frac{\lambda}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_0|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{x} - \boldsymbol{y})\right), \tag{10}$$

where $\mathcal{N}(\boldsymbol{x}; \boldsymbol{m}, \boldsymbol{\Sigma})$ denotes the Gaussian probability density function with mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma}$. Now define $\boldsymbol{\Delta} = \boldsymbol{X} - \boldsymbol{Y} = \boldsymbol{W}\boldsymbol{Z}$, where $\boldsymbol{W}^\top = \begin{pmatrix} \boldsymbol{I} \\ -\boldsymbol{I} \end{pmatrix}$. As $\boldsymbol{W}$ is a linear map, the distribution of $\boldsymbol{\Delta}$ follows easily from Gaussian calculus

$$\boldsymbol{\Delta} \sim \mathcal{N}(\boldsymbol{W}\boldsymbol{m}, \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^\top) = \tag{11}$$

$$\mathcal{N}(\boldsymbol{m}_X - \boldsymbol{m}_Y, \boldsymbol{\Sigma}_{X,X} + \boldsymbol{\Sigma}_{Y,Y} - \boldsymbol{\Sigma}_{X,Y} - \boldsymbol{\Sigma}_{Y,X}). \tag{12}$$

where the indexed $\boldsymbol{m}$ and $\boldsymbol{\Sigma}$ correspond to the appropriate partitions of the mean and covariance matrix. Note that the shapes match as the covariance matrices describe covariances between different dimensions of two points — we are still only considering the kernel similarity between two elements of the RKHS.

Writing the expectation in terms of $\boldsymbol{Z}$ (and using that SE is stationary),

$$k_{\mathrm{E}}(\boldsymbol{X}, \boldsymbol{Y}) = \mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}[k(\boldsymbol{X} - \boldsymbol{Y})] = \tag{13}$$

$$\mathbb{E}_{\boldsymbol{\Delta}}[k(\boldsymbol{\Delta})] = \lambda \, \mathbb{E}_{\boldsymbol{\Delta}}[\mathcal{N}(\boldsymbol{0}; \boldsymbol{\Delta}, \boldsymbol{\Sigma}_0)]\,, \tag{14}$$

we find a Gaussian integral of the type $\int \mathcal{N}(\cdot; \boldsymbol{x}, \boldsymbol{C}_1)\, \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}, \boldsymbol{C}_2)\, \mathrm{d}\boldsymbol{x}$ which results in the analytical kernel expression for the SEEK kernel:

$$k_{\mathrm{SEEK}}(X, Y) = \lambda \mathcal{N}(\boldsymbol{0}; \boldsymbol{W}\boldsymbol{m}, \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^\top + \boldsymbol{\Sigma}_0). \tag{15}$$

If we take X and Y to be one-dimensional random Gaussian variables with joint mean and covariance

$$\boldsymbol{m} = \begin{pmatrix} m_X \\ m_Y \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{X,X} & \sigma_{X,Y} \\ \sigma_{X,Y} & \sigma_{Y,Y} \end{pmatrix}, \tag{16}$$

then we can write the one-dimensional SEEK kernel as

$$k_{\mathrm{SEEK}}(X, Y) = \lambda \mathcal{N}(0; m_{\mathrm{SEEK}}, \sigma^2_{\mathrm{SEEK}}) \tag{17}$$

$$m_{\mathrm{SEEK}} = m_X - m_Y, \tag{18}$$

$$\sigma^2_{\mathrm{SEEK}} = \sigma_0^2 + \sigma_{X,X} + \sigma_{Y,Y} - 2\sigma_{X,Y}. \tag{19}$$

Extending this kernel to multiple dimensions using the typical separable construction of taking tensor products corresponds to assuming that the covariance block $\boldsymbol{\Sigma}_{X,Y}$ (along with the diagonal blocks) in the multi-dimensional setting is diagonal, i.e. there is no covariance across different dimensions.

Inspecting the kernels, we see that as the variance and covariance goes to 0, we recover the original embedding kernel. We also note that on the diagonal the equality between the SEEK and the embedding kernel always holds true as the additional covariance terms cancel appropriately. As $\boldsymbol{\Sigma}_0$ controls the lengthscale of the kernel, we see that it is through modifications of this parameter the input covariance affects the kernel similarity, but we emphasize that the normalization term (see the original SE kernel in equation (9)) means that changes to the covariance will also influence the effective prior variance.

Another point of interest is that while it is useful to consider SEEK as a kernel on the subspace of Gaussians alone, it remains a general E-kernel on the space of all distributions — it is just a happy coincidence that we can evaluate it exactly for Gaussians when using an SE embedding. This means that we can also evaluate the kernel similarity between a Gaussian and e.g. an empirical measure or

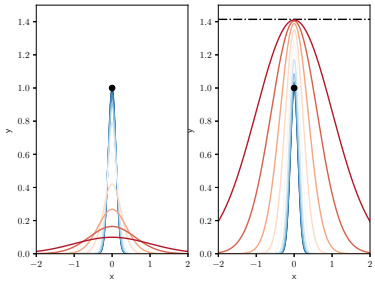


Figure 1: As input noise on the single black observation increases, SEEK (left) deflates and KME (right) inflates. Blue is low input noise, red is high. Top line indicates inflation bound of $\sqrt{2}$.

even a deterministic point in the form of a Dirac delta distribution (both of which can be computed analytically). This is a feature shared with the kernel mean embedding, and KME kernels over empirical measures have already seen some use [Flaxman et al., 2015].

This also helps illustrate the "inflation effect" described in section 2.1.1. Returning to the single-observation scenario proposed there, and further assuming a (KME-)SEEK kernel with lengthscale σ_0^2 and a Gaussian measure on the observed location $X \sim \mathcal{N}(0, \beta\sigma_0^2)$ for some proportionality factor $\beta \geq 0$, then

$$k_{\text{SEEK}}(\delta_0, X) = k_{\text{KME}}(\delta_0, X) = \frac{1}{\sqrt{2\pi\sigma_0^2(1+\beta)}}$$
$$k_{\text{SEEK}}(X, X) = \frac{1}{\sqrt{2\pi\sigma_0^2}},$$
$$k_{\text{KME}}(X, X) = \frac{1}{\sqrt{2\pi\sigma_0^2(1+2\beta)}},$$

for kernels evaluated at the Dirac point measure on zero δ_0, leading to mean predictions at $x_0 = 0$ of

$$\hat{y}_{\text{SEEK}} = \sqrt{\frac{1}{1+\beta}}y, \quad \hat{y}_{\text{KME}} = \sqrt{\frac{1+2\beta}{1+\beta}}y, \tag{20}$$

As the relative variance β increases, we see that SEEK regresses back towards the prior mean, while KME actually inflates its prediction, converging to a $\sqrt{2}$ factor in the limit (see figure 1).

2.3 Beyond Gaussians: Uniform SEEK

While Gaussians appear frequently, E-kernels can be applied beyond that domain. Another common family of distributions is the uniform distribution. If we once again consider the SE kernel of equation (9), but limit it to have diagonal covariance $\boldsymbol{\Sigma}_0$, then we can compute a closed-form kernel over uniform distributions on axis-aligned rectangles $A = [\ell_1, u_1] \times \ldots \times [\ell_D, u_D]$,

$$\mathrm{U}_A(\boldsymbol{x}) = \prod_{d=1}^{D} \frac{\mathbb{1}[\ell_d \leq x_d \leq u_d]}{u_d - \ell_d}. \tag{21}$$

As the kernel factorizes across dimensions, it suffices to consider the one-dimensional kernel as the multivariate version can be constructed from tensor products.

The E-kernel for uniform distributions over $A = [\ell_a, u_a]$ and $B = [\ell_b, u_b]$ over the SE kernel can be computed via the double integral

$$k(\mathrm{U}_A, \mathrm{U}_B) = \frac{\lambda}{|A||B|} \int_{\ell_b}^{u_b} \int_{\ell_a}^{u_a} \mathcal{N}(x; y, \sigma_0^2)\, \mathrm{d}x\, \mathrm{d}y\,. \tag{22}$$

We recognize the inner-most integral as being computable in terms of the Gaussian standard CDF Φ as $\Phi(\frac{u_a-y}{\sigma_0}) - \Phi(\frac{\ell_a-y}{\sigma_0})$. The outer integral is then an integral over the Gaussian CDF, with the antiderivative of the CDF being,

$$\bar{\Phi}(t,y) = \int \Phi\left(\frac{t-y}{\sigma_0}\right) \mathrm{d}y = \tag{23}$$

$$(y-t)\Phi\left(\frac{t-y}{\sigma_0}\right) - \sigma_0^2 \mathcal{N}\left(y; t, \sigma_0^2\right). \tag{24}$$

The kernel is then

$$k(\mathrm{U}_A, \mathrm{U}_B) = \frac{\lambda}{|A||B|}((\bar{\Phi}(u_a,u_b) - \bar{\Phi}(u_a,\ell_b)) - \tag{25}$$

$$(\bar{\Phi}(\ell_a,u_b) - \bar{\Phi}(\ell_a,\ell_b))). \tag{26}$$

We name this the Uniform SEEK kernel, or USEEK for short.

2.4 Empirical E-kernels

E-kernels are only analytical when the necessary double integrals can be carried out, as in the above two cases. For many combinations of kernel and probability distribution, this will not be possible. One approximate way of tackling this for an arbitrary kernel and an arbitrary joint distribution p_{XY} is to do a sample approximation of the distribution using S samples:

$$\mathbb{E}[k(X,Y)] \approx \frac{1}{S}\sum_{i=1}^{S} k(x_i, y_i), \quad x_i, y_i \sim p_{XY}. \tag{27}$$

This Monte Carlo approximation can be made arbitrarily precise, by simply using sufficiently many samples. These empirical E-kernels can also be used jointly with MCMC algorithms, to carry out regressions on latent variables.

2.5 Nested Gaussian Process

To demonstrate the E-kernel's ability to model sets of mutually covariant random variables in a way that is outside the scope of the KME kernels, we will here use a Gaussian process to model the input uncertainties, demonstrating how we can form a new kernel by calculating the E-kernel of the Gaussian process.

As the Gaussian process is simply multivariate Gaussian for any finite set of observations, it falls neatly into the SEEK framework. Limiting ourselves to the one-dimensional setting without loss of generality, we take $x, y \in \mathbb{R}$ to be two points in the input space. Given an input kernel k, define auxiliary points $\hat{X}$ and $\hat{Y}$ as being generated by a Gaussian process with input covariance k

$$\begin{pmatrix}\hat{X}\\ \hat{Y}\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}x\\ y\end{pmatrix}, \begin{pmatrix}k(x,x) & k(x,y)\\ k(y,x) & k(y,y)\end{pmatrix}\right). \tag{28}$$

Here we have set the mean function to be the identity $m(x) = x$, reflecting that we are modeling the noise around a mean position x. To get the E-kernel, we then simply have to replace $\boldsymbol{m}$ and $\boldsymbol{\Sigma}$ in equations (16) and (17) by the above mean and covariance functions of the Gaussian process, evaluated at input points x and y. We call this the nested Gaussian process (NGP) kernel, and it's defined as $k_{NGP}(x,y) \equiv k_{\mathrm{SEEK}}(\hat{X}, \hat{Y})$, and more explicitly as

$$k_{NGP}(x,y) = \lambda \mathcal{N}\left(0; x-y, \sigma_0^2 + \|x-y\|^2_{\mathcal{H}_k}\right), \tag{29}$$

by plugging into equation (17), where we have introduced the kernel norm,

$$\|x-y\|^2_{\mathcal{H}_k} = k(x,x) + k(y,y) - k(x,y) - k(y,x), \tag{30}$$

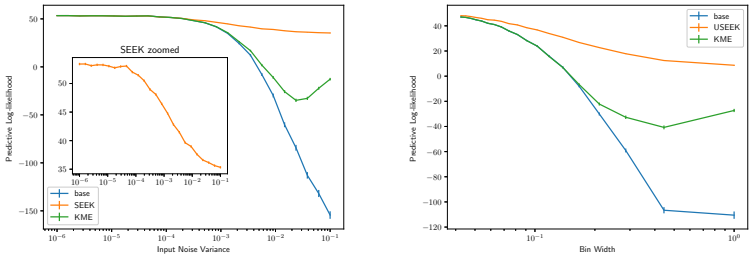
to simplify the expression.

If the same construction was attempted using a kernel mean embedding construction, the covariance terms would disappear and we would be left with,

$$k_{KME-NGP}(x,y) = \lambda \mathcal{N}\left(0; x-y, \sigma_0^2 + 2k(0,0)\right), \tag{31}$$

taking the marginal variance into account and little else.

We leave this kernel design as a curiosity



(a) SEEK's generalization performance when observations have Gaussian input noise. The inserted axis shows SEEK's performance decline with a more appropriate y-axis.

(b) USEEK's generalization performance as a function of the bin width of the censoring discretization.

Figure 2: The sample mean and its standard deviation for the predictive log-likelihood on the test set for 1000 repeat experiments.

3 Demonstrations

In this section we will briefly demonstrate how the SEEK and USEEK kernels lead to qualitatively different behavior compared to the standard kernel mean embedding kernels.

3.1 SEEK kernel

The SEEK kernel allows us to build regressions on spaces of Gaussian random variables and/or mixed spaces of Gaussians, points, and empirical measures (or even Gaussian mixtures).

We will demonstrate the SEEK on independent Gaussians, meaning that it will only differ from the KME kernel on the diagonal as previously noted. We offer two small demonstrations to show off some of the kernel's properties.

3.1.1 Noisy Input

One of the proposed uses of SEEK is as a kernel for regression with noisy inputs. To demonstrate this functionality, we generate a grid of $N_{train} = 10$ training points on the interval $[-1, 1]$ and perturb it 1000 times by Gaussian random noise for 25 different noise variances between 10^{-6} and 10^{-1}. For each of these experiments we construct a random function by sampling a Gaussian process at the N_{train} perturbed training points as well as $N_{test} = 100$ test points. We use a squared exponential kernel with a lengthscale of 0.1 and a prior variance of 1. We also add Gaussian noise with variance 10^{-2} to the observations. We use x_i to denote the input locations, $\tilde{x}_i$ to denote the input with input noise added, $y_i = f(\tilde{x}_i)$ to denote the evaluation of the latent function at the perturbed location, and $\tilde{y}_i$ to denote the output with output noise. Now assume that the unperturbed training data locations, the function evaluations at the perturbed training data locations, the input noise variance, and the kernel parameters are available to us. Our naive baseline is a Gaussian process with a squared exponential kernel using the true kernel parameters and trained on $(x_i, \tilde{y}_i)$, i.e. the model is misspecified due to neglecting the input noise. We compare with a GP trained with both the SEEK and the KME kernel on Gaussian inputs $\mathcal{N}(x_i, \nu)$ where ν is the known input variance. We assume the test points are deterministic with input variance 0, and report the mean (along with standard errors) of the predictive log-likelihood over the many repeat experiments. We see in figure 2a that despite both models having full knowledge of the process parameters, SEEK starts to outperform the standard Gaussian model which assumes no input noise, as well as the KME kernel. As such the problem is one of model misspecification, which SEEK seems to adjust for quite well. Ultimately, the performance boost comes from increased smoothing and noise, so the classical kernel could potentially be made competitive by changing the length-scale and output noise, but given knowledge of the generative model parameters it's not clear how to introduce the input variance into the standard SE kernel in an appropriate manner.

3.1.2 Censored Input

Using the USEEK kernel, we can handle cases where the data was censored for privacy reasons or due to missing information or partial observations. In these scenarios, data consists of intervals containing the true data point. We construct a scenario with censored data by generating 50 data points and then discretizing the points by identifying them with one of K evenly spaced bins over the interval $[-2, 2]$. We train USEEK and its KME variant by taking these intervals to be uniform distributions over the point's potential location, and we train the baseline Gaussian process by identifying all data points with the mean of their assigned interval, as a best guess of the true point's location.

Following the experimental paradigm of section 3.1.1 and averaging over 1000 generated datasets using the same kernel parameters, we get the result depicted in figure 2b. The predictive likelihood decreases as the bins grow bigger, but the rate of decrease is significantly different for the three methods, with USEEK demonstrating the slowest decrease and overall best performance.

4 Related Methods

The Joint Expectation Kernel and the KME kernel draw connections to a number of existing methodologies.

The idea of making the standard squared exponential kernel non-stationary by allowing the lengthscale to vary locally was treated thoroughly by Paciorek in e.g. [2003, 2006], but as they make clear you cannot just plug an arbitrary function in place of the lengthscale parameter and assume the function is PSD. We note that their construction bears resemblance to a KME kernel with a white noise embedding kernel.

Regression on distributions has been pursued by multiple authors using the KME kernel, typically on empirical measures where there is a computational challenge as N atoms in the measure leads to a computational cost scaling like K^2 as all atoms have to be compared pairwise. [Flaxman et al., 2015, Póczos et al., 2013, Oliva et al., 2014]

Uncertain inputs have also been treated using variational inducing points approximations, which of course necessitates solving the variational problem which is the primary limitation [Damianou et al., 2016].

5 Conclusion

In this paper we have described a new family of kernels in the form of the joint expectation kernels, or E-kernels, and shown how they extend the kernel mean embedding concept.

We have pointed out some of the key discrepancies between the classical KME kernel and the E-kernels, such as the diagonal deviation, the different limit behavior, and the inflation effect. We argue that E-kernels are more suitable than KME kernels for domains of random variables whereas KME is superior for domains of marginal probability distributions or integrals.

Furthermore, we have detailed the construction of two analytical kernels in the form of the SEEK kernel for regression on Gaussians and the USEEK for regression on uniform interval distributions. We also described a new way of constructing novel kernels by applying the E-kernel to a Gaussian process, resulting in the nested Gaussian process.

References

Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322, 2002.

Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2990–2998. Curran Associates, Inc., 2016.

Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems*, pages 625–632, 2001.

Eleazar Eskin, Jason Weston, William S. Noble, and Christina S. Leslie. Mismatch string kernels for SVM protein classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1441–1448. MIT Press, 2003.

Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 10–18. Curran Associates, Inc., 2012.

Seth R. Flaxman, Yu-Xiang Wang, and Alexander J. Smola. Who supported obama in 2012?: Ecological inference through distribution regression. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 289–298, New York, NY, USA, 2015. ACM.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. May 2016.

Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In Marcus Hutter, Rocco A. Servedio, and Eiji Takimoto, editors, *Algorithmic Learning Theory*, volume 4754 of *Lecture Notes in Computer Science*, pages 13–31. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12:2389–2410, February 2011.

François-Xavier Briol, Chris J. Oates, Mark Girolami, Michael A. Osborne, and Dino Sejdinovic. Probabilistic integration: A role in statistical computation? December 2015.

George Casella and Roger L. Berger. *Statistical Inference*. Thomson Learning, 2nd edition, 2002.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, August 2006.

Ryan Darby Turner. *Gaussian Processes for state space models and change point detection*. PhD thesis, University of Cambridge, 2012.

Christopher J. Paciorek. Nonstationary Gaussian processes for regression and spatial modelling. 2003.

Christopher J. Paciorek and Mark J. Schervish. Spatial modelling using a new class of nonstationary covariance functions. *Environmetrics*, 17(5):483–506, 2006.

B Póczos, A Singh, A Rinaldo, and L A Wasserman. Distribution-Free distribution regression. *AISTATS*, 2013.

Junier Oliva, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Fast distribution to real regression. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 706–714, 2014.

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: JMLR*, 17(42):1–62, 2016.

A.3 Matrix Product states for inference in discrete probabilistic models

The following paper is yet to be published. It has been submitted to JMLR. It is related to background material covered in chapter 4, but also to parts of chapter 2.

Matrix Product States for inference in discrete probabilistic models

Rasmus Bonnevie rabo@dtu.dk
Cognitive Systems, Department of Applied Mathematics and Computer Science
Technical University of Denmark
2800 Kgs. Lyngby, Denmark

Mikkel N. Schmidt mnsc@dtu.dk
Cognitive Systems, Department of Applied Mathematics and Computer Science
Technical University of Denmark
2800 Kgs. Lyngby, Denmark

Editor:

Abstract

When faced with problems involving inference in discrete domains, solutions often involve appeals to conditional independence structure or mean-field approximations. We argue that this is insufficient for a number of interesting Bayesian problems, including mixture assignment posteriors and probabilistic relational models (e.g. the stochastic block model). These posteriors exhibit no conditional independence structure, precluding the use of graphical model methods, yet exhibit dependency between every single element of the posterior, making mean-field methods a poor fit. We propose using an expressive yet tractable approximation inspired by tensor factorization methods, alternately known as the tensor train or the matrix product state, and which can be construed of as a direct extension of the mean-field approximation to higher-order dependencies. We illustrate how to efficiently perform marginalization, conditioning, sampling, normalization, some expectations, and approximate variational inference in our proposed model.

Keywords: variational inference, matrix product states, tensor trains, discrete models, symmetry

1. Introduction

Inference in discrete Bayesian probabilistic models has been studied intensely for decades. In terms of a graphical model, inference problems can be solved exactly based on dynamic programming such as the junction tree algorithm (Lauritzen and Spiegelhalter, 1988; Wainwright and Jordan, 2008); however, the computational cost scales exponentially in the so-called treewidth, intuitively a measure of graph connectedness, roughly implying that sparse graphs (few dependencies) are straightforward to perform inference on, while densely connected graphs can be computationally intractable. Unfortunately, for many hierarchical models such as mixture models and probabilistic relational models, marginalization of nuisance parameters leads to posteriors that are completely connected, meaning that algorithms which rely on local conditional independence fall short.

One solution has been to use Markov chain Monte Carlo (MCMC) sampling, and in many scenarios marginalization actually induces posteriors that are more easily traversed

by MCMC algorithms as the nuisance parameters no longer have to be set appropriately for a particular configuration to be likely (Teh et al., 2007). But MCMC on discrete spaces also prohibits the use of efficient gradient-based samplers and it Gibbs sampling and other MCMC methods often tend to get stuck in local modes of the posterior distribution.

Variational inference is a completely separate strategy which has been applied successfully to e.g. mixture inference (Bishop, 2006; Hughes and Sudderth, 2013). The idea is to form an analytic approximation of the posterior distribution by minimizing a statistical distance between some tractable family of distributions and the true posterior. One issue with variational inference we would like to highlight here is that the approximations are often quite limited in their expressiveness, and suffer more from mode collapse than even the Gibbs sampler, as we will illustrate.

We consider probabilistic joint models of the form $p(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{Y}$ is a set of observed random variables (discrete and/or continuous) and $\boldsymbol{X}$ is a set of latent discrete random variables. We assume that there is no exploitable conditional independence structure in the model. Let N denote the number of latent variables $\boldsymbol{X} = \{X_n\}_{n=1}^N$, each of which take values in a discrete space $\mathcal{X}_n$ of size K_n. We denote every set $\boldsymbol{x} = \{x_i\}_{n=1}^N$ where $x_n \in \mathcal{X}_n$ a configuration of the random variables, and we note that there are $K^* = \prod_{n=1}^N K_n$ different discrete configurations.

Inference is the procedure of reasoning appropriately given observations, which in the Bayesian setting involves evaluating as well as computing marginals and expectations over the posterior $p(\boldsymbol{X}|\boldsymbol{Y})$. This is computationally intractable in most cases, since it involves evaluating the model evidence $p(\boldsymbol{Y})$: a sum with an exponential number of terms. Variational inference circumvents the problem by defining a variational approximation $q(\boldsymbol{x};\boldsymbol{\theta})$ to the true posterior $p(\boldsymbol{x}|\boldsymbol{Y})$ and maximizing the so-called evidence lower bound objective (ELBO)

$$\ln p(\boldsymbol{Y}) \geq \mathcal{L} = \mathbb{E}[\ln p(\boldsymbol{x}, \boldsymbol{Y})] - \mathbb{E}[\ln q(\boldsymbol{x};\boldsymbol{\theta})], \tag{1}$$

the optimal solution of which also implies the approximation with the lowest KL divergence to the posterior (Wainwright and Jordan, 2008). While recent innovations have extended the tractable classes of approximations for probabilistic models over continuous random variables to arbitrarily complex distributions (Kingma and Welling, 2014; Ranganath et al., 2014; Rezende and Mohamed, 2015), for most discrete distributions it is very uncommon to see approximations that are not mean-field, i.e., where the approximation factorizes completely as $q(\boldsymbol{x};\boldsymbol{\theta}) = \prod_{n=1}^N q(x_n;\boldsymbol{\theta}_n)$. In the discrete case, this leaves a rather constrained design space as each independent discrete factor can in all generality be modeled with a categorical distribution represented by a probability vector $\boldsymbol{\theta}_n$, where $q(x_n = k) = \theta_{n,k}$ and $\sum_{k=1}^{K_n} \theta_{n,k} = 1,\ \forall n$.

While this approximation has been used successfully to find clusterings (Hughes and Sudderth, 2013), topics (Teh et al., 2007), and communities (Xu et al., 2014), it does so in part by being exceedingly coarse. It is well-known that approximations found through variational inference tend to underestimate variance and are mode-seeking by nature (Minka, 2005), and in the discrete setting this often translates into a low-variance collapse unto a particular locally-optimal configuration $\boldsymbol{x}^*$.

2. Probability Tensor Decomposition

To get a sense for the low fidelity of the mean-field approximation, we will draw a connection between multivariate discrete distributions and tensors. We will follow Kolda and Bader (2009) in giving a brief outline of tensors and operations thereon. An N'th order (or N-way) tensor is a multidimensional array $\mathcal{T}$ of shape $K_1 \times \ldots \times K_N$ where the element indexed by $\mathcal{I} = (i_1, \ldots, i_N)$ is denoted by $\mathcal{T}_{\mathcal{I}}$. Rows and columns generalize to mode-n fibers, defined as the vector $\boldsymbol{v}_{i_1,\ldots,i_{n-1},i_{n+1},\ldots i_N}$ where the elements are taken from a slice of the tensor where every index but one is kept fixed, i.e. $[\boldsymbol{v}_{i_1,\ldots,i_{n-1},i_{n+1},\ldots i_N}]_{i_n} = \mathcal{T}_{i_1,\ldots,i_N}$. To relate tensors to matrices, we can perform a matricization of $\mathcal{T}$ on mode n, denoted by $\boldsymbol{T}_{(n)}$, by stacking all of its mode-n fibers row-wise into a matrix of shape $(\prod_{m\neq n} K_m) \times K_n$.[1] Finally, we have the n-mode product $\times_n$ of a tensor with a matrix operator which we can conveniently express in terms of the matricized tensor

$$(\mathcal{T} \times_n \boldsymbol{A})_{(n)} = \boldsymbol{A}\boldsymbol{T}_{(n)}. \tag{2}$$

There are two notions of rank, each going hand-in-hand with a particular kind of tensor decomposition. The first is the tensor rank R, which is the minimal number of vector outer products (otherwise known as rank-one tensors) needed to sum to the tensor $\mathcal{T}$. It is related to the canonical polyadic (CP) decomposition of the form

$$\mathcal{T} = \sum_{r=1}^{R} \boldsymbol{v}_1^{(r)} \circ \ldots \circ \boldsymbol{v}_N^{(r)}, \tag{3}$$

where $\circ$ is the tensor outer product such that $[\mathcal{T} \circ \boldsymbol{v}]_{i_1,\ldots,i_{D+1}} = \mathcal{T}_{i_1,\ldots,i_D} v_{i_{D+1}}$. Second, there's the multilinear rank which is a vector $(r_1, \ldots, r_N)$, where r_n is equivalently the rank of $\boldsymbol{T}_{(n)}$ or the minimal number of rows in the matrix $\boldsymbol{U}_n$ which features in the higher-order SVD (or Tucker) decomposition

$$\mathcal{T} = \mathcal{C} \times_1 \boldsymbol{U}_1 \times_2 \ldots \times_N \boldsymbol{U}_n, \tag{4}$$

where $\mathcal{C}$ is the $r_1 \times \ldots \times r_N$ core tensor. By counting the number of rank-one terms in the Tucker decomposition, we get that $R \leq \prod_{n=1}^{N} r_n \leq \prod_{n=1}^{N} K_n$. It should be noted that the final upper bound is loose.

2.1 Probability Tensors

To connect this to probabilistic models, consider that we can associate each configuration $\boldsymbol{x}$ with a posterior probability value $p(\boldsymbol{x}|\boldsymbol{Y})$. We can combine these values into a $K_1 \times \ldots \times K_N$ *probability tensor* $\mathcal{T}_{\boldsymbol{x}} = p(\boldsymbol{x}|\boldsymbol{Y})$, where the $\boldsymbol{x}$ subscript indicates that dimension n is indexed by the value of x_n. This is a probability tensor in the sense that each element corresponds to a configuration, and it sums to 1, so if we vectorized it into a vector of length K^*, it could parameterize a categorical distribution over all possible configurations, similarly to the $\boldsymbol{\theta}_n$ parameters described previously.

Now, as a distribution, the mean-field approximation $q(\boldsymbol{x}; \boldsymbol{\theta})$ also defines a probability tensor with elements

$$\hat{\mathcal{T}}_{\boldsymbol{x}} = \theta_{1,x_1} \ldots \theta_{N,x_N}. \tag{5}$$

1. the stacking order is inconsequential, as long as it is consistent.

In tensor parlance, this is a rank-one tensor of form $\hat{\mathcal{T}} = \boldsymbol{\theta}_1 \circ \ldots \circ \boldsymbol{\theta}_N$. Returning to the concept of tensor rank, we note that the approximation has the minimal rank possible, and is thus in some sense maximally simple. It seems optimistic to believe that the posterior probability tensors will have such simplistic structure, and Kolda and Bader (2009) go on to cite a result of a Monte Carlo experiment demonstrating that even for a small randomly-generated $2 \times 2 \times 2$ tensor, rank-one tensors occur with zero probability.[2]

2.2 Tensor Trains

So if we accept that approximation with a rank-one tensor is a flawed approach, what should we do instead? Recall that the sought tensor $\mathcal{T}$ has $K^* = \prod_{n=1}^{N} K_n$ elements, which grows exponentially with N, which rules out a naive representation. We already saw two structured representations, namely the CP representation of equation (3), and the Tucker representation in (4). Tucker unfortunately suffers from the same combinatorial explosion as $\mathcal{T}$ due to the existence of the tensor $\mathcal{C}$. CP on the other hand is convenient, but given that expressing $\mathcal{T}$ can require up towards K^* rank-one tensors, it might not be the most parsimonious representation.

Oseledets (2011) propose a different decomposition. Starting from the tensor $\mathcal{T}$, we can find a low-rank decomposition of the mode-1 unfolding as

$$\boldsymbol{T}_{(1)} = \boldsymbol{U}_1 \boldsymbol{V}_1^\top, \tag{6}$$

where we choose $\boldsymbol{U}_1$ to be orthogonal (which is possible using SVD). Now, $\boldsymbol{V}_1$ will have shape $r_1 \times \prod_{n=2}^{N} K_n$ where r_1 is the rank of the low-rank decomposition used. To continue, we reshape $\boldsymbol{V}_1$ to have shape $r_1 K_2 \times \prod_{m=3}^{N} K_m$. We can then recursively define $\boldsymbol{U}_n$ as

$$\boldsymbol{U}_n \boldsymbol{V}_n^\top = \text{Reshape}\left(\boldsymbol{V}_{n-1};\ (r_{n-1}K_n,\ \prod_{m=n+1}^{N} K_m)\right). \tag{7}$$

Compared to CP and Tucker, it is less clear how to expand this into a tensor expression. But if we define the so-called core tensors $\mathcal{G}_n = \text{Reshape}(\boldsymbol{U}_n;\ (r_{n-1}, K_n, r_n))$ with $r_0 = r_N = 1$, then we can elegantly express the individual indices of the tensor as

$$\mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1]\mathcal{G}_2[i_2]\ldots\mathcal{G}_N[i_N], \tag{8}$$

where we abuse notation slightly to define $[\mathcal{G}_n[k]]_{ij} = [\mathcal{G}_n]_{ikj}$ to be the 2nd mode matrix slices. This sequential construction of "carriages" linked together has led to naming the decomposition the tensor train (TT) decomposition.

If we insist on a formula similar to equation (3) or (4), we can define the block vec product as

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \boxtimes \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = \begin{pmatrix} A_1 B_1 \\ A_1 B_2 \\ A_2 B_1 \\ A_2 B_2 \end{pmatrix}, \tag{9}$$

block vec product

2. note, though, that the posterior probability tensors are not random draws from the space of tensors.

following which we can define the vectorized TT-tensor as (Van Loan, 2008)

$$\operatorname{vec}(\mathcal{T}) = \tilde{\boldsymbol{U}}_1 \boxtimes \ldots \boxtimes \tilde{\boldsymbol{U}}_N, \quad \tilde{\boldsymbol{U}}_n = \begin{pmatrix} \mathcal{G}_n[1] \\ \vdots \\ \mathcal{G}_n[K_n] \end{pmatrix}. \tag{10}$$

Evaluation of the tensor train requires N matrix products, which would normally cost $\mathcal{O}(r_n^3)$ a piece, but since the first and last core have vector-shaped slices, we can calculate the whole train using only matrix-vector products ($\mathcal{O}(r_n^2)$) by starting multiplication from the left or the right.

The central advantage of tensor trains is the number of operations that can be efficiently implemented directly on the representation (Oseledets, 2011):

Contraction If we want to sum out index i_n, we can write the tensor in terms of its indices and move the sum to the appropriate matrix core

$$\sum_{i_n=1}^{K_n} \mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1] \ldots \left(\sum_{i_n=1}^{N} \mathcal{G}_n[i_n] \right) \ldots \mathcal{G}_N[i_N]. \tag{11}$$

Scaling Scaling every core $\mathcal{G}_n$ by α_n is equivalent to scaling the tensor train by $\prod_{n=1}^{N} \alpha_n$.

Addition As can be verified using standard linear algebra, the TT decomposition $\mathcal{C}_n$ of the addition of two TT-tensors with cores $\mathcal{A}_n$ and $\mathcal{B}_n$ can be found to be

$$\mathcal{C}_1[k] = \begin{pmatrix} \mathcal{A}_1[k] & \mathcal{B}_1[k] \end{pmatrix}, \quad \mathcal{C}_N[k] = \begin{pmatrix} \mathcal{A}_N[k] \\ \mathcal{B}_N[k] \end{pmatrix}, \tag{12}$$

$$\mathcal{C}_n[k] = \begin{pmatrix} \mathcal{A}_n[k] & \boldsymbol{0} \\ \boldsymbol{0} & \mathcal{B}_n[k] \end{pmatrix}, \quad n \neq 1 \wedge n \neq N. \tag{13}$$

Multiplication Finally, the TT decomposition $\mathcal{C}_n$ of the element-wise multiplication of two TT-tensors $\mathcal{A}$ and $\mathcal{B}$ with cores $\mathcal{A}_n$ and $\mathcal{B}_n$ is simply $\mathcal{C}_n[k] = \mathcal{A}_n[k] \otimes \mathcal{B}_n[k]$, where $\otimes$ is the Kronecker product, as follows from

$$\mathcal{A}_{\mathcal{I}}\mathcal{B}_{\mathcal{I}} = \operatorname{Tr} \mathcal{A}_{\mathcal{I}} \operatorname{Tr} \mathcal{B}_{\mathcal{I}} = \operatorname{Tr}[\mathcal{A}_{\mathcal{I}} \otimes \mathcal{B}_{\mathcal{I}}] = \operatorname{Tr}[(\mathcal{A}_1[i_1] \otimes \mathcal{B}_1[i_1]) \ldots (\mathcal{A}_N[i_N] \otimes \mathcal{B}_N[i_N])], \tag{14}$$

These operations are sufficient for e.g. calculating the Frobenius norm of a tensor (Oseledets, 2011), but they also turn out to be extremely useful for probabilistic models as we will now show.

Just like in algebra, there are a number of operations that are essential to probability theory, and most of them turn out to be quite easy to carry out on a probability tensor in TT format:

Marginalization We can find any desired marginal distribution by applying contraction to the indices that we want to marginalize over. As a convenience, we define the marginal core matrices

$$\boldsymbol{M}_n = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n], \tag{15}$$

so that marginalization can be written as

$$p(\boldsymbol{x}_{/n}) = \sum_{i_n=1}^{K_n} \mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1] \dots \boldsymbol{M}_n \dots \mathcal{G}_N[i_N], \tag{16}$$

where $\boldsymbol{x}_{/n}$ denotes the full set of random variables $\{x_n\}_{n=1}^N$, excluding the n'th element.

Normalization Given an unnormalized probability tensor (i.e. any non-negative tensor), we can easily calculate the normalizing constant of the tensor by using the contraction operation presented above on every index. As contraction is the same as marginalization, we can write the normalization constant using the marginal cores defined above

$$Z = \boldsymbol{M}_1 \dots \boldsymbol{M}_N. \tag{17}$$

Having computed the normalization constant, we can also normalize the tensor using the scaling operation.

Conditioning Conditional distributions are likewise easily computed by fixing the core slices corresponding to the observations, and marginalizing out all remaining indices to compute the normalization constant.

As a further note of interest, we can also compute a large class of expectations, namely all those where the quantity of interest can be written as another tensor train $\mathcal{Q}$. The expectation is then simply

$$\mathbb{E}[\mathcal{Q}_I] = \sum_{\mathcal{I}} \mathcal{T}_{\mathcal{I}} \mathcal{Q}_{\mathcal{I}}, \tag{18}$$

which is an element-wise product, followed by a complete contraction over all indices.

The main limitations of tensor trains is picking ranks, ordering the dimensions, and ensuring non-negativity. The rank of the tensor train is a fundamental hyperparameter which will effectively correspond to the expressiveness of the approximation. While most tensors seem to not be full rank, it makes sense to choose this parameter based predominantly on computational budget. There are no proper guidelines for ordering the dimensions, although the result saying that a perfect approximation exists holds no matter the ordering. It's also possible to use a tensor ring decomposition instead, which makes the tensor train invariant to cyclic permutations of the dimensions (Zhao et al., 2016). Non-negativity is the most crucial problem, as it is a necessary constraint which has to hold globally.

By simple linear algebra arguments, it is clear that if all the cores are non-negative, then the tensor will be non-negative as well. From a tensor decomposition point of view it is however unlikely that the best representation of the tensor has only non-negative cores. Another construction that ensures non-negativity is a squaring, such that the tensor train $\hat{\mathcal{T}}$ itself models the square root of the probability tensor. By the algebra rules presented above, this implies a tensor train

$$\mathcal{T}_{\mathcal{I}} = \hat{\mathcal{T}}_{\mathcal{I}}^2 = (\hat{\mathcal{G}}_1[i_1] \otimes \hat{\mathcal{G}}_1[i_1]) \dots (\hat{\mathcal{G}}_N[i_N] \otimes \hat{\mathcal{G}}_N[i_N]). \tag{19}$$

That is, the probability tensor can be guaranteed to be positive if each core matrix of the probability tensor can be represented by a Kronecker product of a matrix with itself,

$$\mathcal{G}_n[i_n] = \hat{\mathcal{G}}_n[i_n] \otimes \hat{\mathcal{G}}_n[i_n]. \tag{20}$$

2.3 Matrix Product State

Interestingly, the tensor train decomposition has been developed independently within the quantum mechanics community under the name of matrix product states (see Schollwöck (2011) for a review of the history of its development). It falls within the larger umbrella of tensor networks, a general framework for constructing arbitrarily complex hierarchical tensor decompositions.

The matrix product state (MPS) literature contains a number of results not present in the tensor train literature, which motivates its introduction here. In quantum mechanics, the MPS $\mathcal{T}$ represents the quantum mechanical wave function of a combined quantum state of N particles where each particle x_n can be in one of K_n states. Following the probabilistic interpretation of quantum mechanics via Born's rule, the probability of the system being in a configuration $\boldsymbol{x}$ is exactly the square of the MPS, similarly to the squaring construction we employed above to ensure non-negativity. While an MPS is really synonymous with a tensor train, we will use the term to describe squared tensor trains as probabilistic models, in contrast with the more general tensor train. A problem with the squaring construction as presented, is that it incurs a bit of a performance hit since we have to operate with cores with squared ranks r_n^2 despite only being able to model the complexity of $\hat{\mathcal{T}}$, the square root of the tensor, with an effective rank of r_n. The articles on MPS propose a solution for this, in the form of the *bubbling algorithm*, which can speed up the computation of contractions (and thus normalization, marginalization, and conditioning) significantly for particular cases (Bridgeman and Chubb, 2017; Robeva and Seigal, 2018).

Instead of following the derivation of multiplication in equation (14), consider the scenario where we want to contract a product of two tensor trains over i_1:

$$\sum_{i_1=1}^{K_1} \mathcal{A}_1[i_1] \dots \mathcal{A}_N[i_N] \mathcal{B}_1[i_1] \dots \mathcal{B}_N[i_N] = \mathcal{A}_N[i_N]^\top \dots \left(\sum_{i_1=1}^{K_1} \mathcal{A}_1[i_1]^\top \mathcal{B}_1[i_1] \right) \dots \mathcal{B}_N[i_N]. \quad (21)$$

Note that by transposition (possible since the quantity is scalar) we have gathered the factors depending on i_1 together. We can then analytically marginalize them out. Instead of being forced to instantiate Kronecker products of the cores, as would be necessary when calculating marginals of equation (19) so that a full contraction costs $\mathcal{O}(Nr^4)$, the bubbling algorithm only requires $\mathcal{O}(2Nr^3)$ operations. If we then want to also marginalize over i_2 we can write

$$\sum_{i_2=1}^{K_2} \mathcal{A}_2[i_2]^\top \left(\sum_{i_1=1}^{K_1} \mathcal{A}_1[i_1]^\top \mathcal{B}_1[i_1] \right) \mathcal{A}_2[i_2]. \quad (22)$$

Returning to our squared tensor trains with cores $\mathcal{G}_n$, we introduce the left marginal matrices recursively as,

$$\boldsymbol{L}_1 = \sum_{i_1=1}^{K_1} \mathcal{G}_1[i_1]^\top \mathcal{G}_1[i_1], \quad \boldsymbol{L}_n = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]^\top \boldsymbol{L}_{n-1} \mathcal{G}_n[i_n], \quad (23)$$

so the contraction over indices $i_1, \dots, i_m$ can be written as

$$\sum_{i_1,\dots,i_m} \hat{\mathcal{T}}_{\mathcal{I}}^2 = \mathcal{G}_N[i_N]^\top \dots \mathcal{G}_{m+1}[i_{m+1}]^\top \boldsymbol{L}_m \mathcal{G}_{m+1}[i_{m+1}] \dots \mathcal{G}_N[i_N]. \quad (24)$$

This is helpful, but only for a rather limited set of contractions and marginals. Fortunately, we can define a similar series of *right* marginal matrices, by first noting that equation (24) can be written as

$$\mathrm{Tr}[\boldsymbol{L}_m \mathcal{G}_{m+1}[i_{m+1}] \dots \left(\mathcal{G}_N[i_N]\mathcal{G}_N[i_N]^\top\right) \dots \mathcal{G}_{m+1}[i_{m+1}]^\top] \tag{25}$$

If we then marginalize over first i_N, then i_{N-1} and so on, we can define the right marginal matrices as

$$\boldsymbol{R}_N = \sum_{i_N=1}^{K_N} \mathcal{G}_N[i_N]\mathcal{G}_N[i_N]^\top, \quad \boldsymbol{R}_n = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]\boldsymbol{R}_{n+1}\mathcal{G}_n[i_n]^\top \tag{26}$$

and then finally, the contraction over $i_1, \dots, i_\ell$ and $i_u, \dots, i_N$ can be written as,

$$\sum_{i_1,\dots,i_\ell} \sum_{i_u,\dots,i_N} \hat{\mathcal{T}}_{\mathcal{I}}^2 = \mathrm{Tr}[\boldsymbol{L}_\ell \mathcal{G}_{\ell+1}[i_{\ell+1}] \dots \mathcal{G}_{u-1}[i_{u-1}] \boldsymbol{R}_u \mathcal{G}_{u-1}[i_{u-1}]^\top \dots \mathcal{G}_{\ell+1}[i_{\ell+1}]^\top] \tag{27}$$

As a future convenience, let G_a^b denote the consecutive product of cores from $\mathcal{G}_a[i_a]$ to $\mathcal{G}_b[i_b]$, then we can also write the above statement as,

$$\sum_{i_1,\dots,i_\ell} \sum_{i_u,\dots,i_N} \hat{\mathcal{T}}_{\mathcal{I}}^2 = \mathrm{Tr}[\boldsymbol{L}_\ell G_{\ell+1}^{u-1} \boldsymbol{R}_u {G_{\ell+1}^{u-1}}^\top]. \tag{28}$$

While this allows us to compute many marginals efficiently, a small caveat is that the efficiency drops once we start to consider non-consecutive marginals since we can only use the recursive formulas on the first and last indices. A final note is that we get a series of identities relating the left and right marginal matrices to the partition function,

$$Z = \mathrm{Tr}[\boldsymbol{L}_n \boldsymbol{R}_{n+1}] = \boldsymbol{L}_N = \boldsymbol{R}_1. \tag{29}$$

2.4 Canonical Representation

Another key feature employed in the MPS literature is the idea of a canonical set of cores. As presented, the tensor train is not a unique representation as applying a so-called gauge transform

$$\tilde{\mathcal{G}}_n[i_n] = \boldsymbol{A}_n^{-1} \mathcal{G}_n[i_n] \boldsymbol{A}_{n+1}, \tag{30}$$

gauge transform

does not change the value of the corresponding tensor, i.e., the tensors represented by $\mathcal{G}_n$ and $\tilde{\mathcal{G}}_n$ are identical for all choices of $\boldsymbol{A}_n$. We can use this to strategically pick cores with desirable properties.

Left- and right-canonical matrix product states constitute two classical forms of canonicity (Schollwöck, 2011), defined as,

$$\sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]^\top \mathcal{G}_n[i_n] = \boldsymbol{I}, \quad \forall n, \tag{31}$$

left-canonical

$$\sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n] \mathcal{G}_n[i_n]^\top = \boldsymbol{I}, \quad \forall n, \tag{32}$$

right-canonical

Note that for left-canonical MPS's, this implies that $\boldsymbol{L}_1 = \boldsymbol{I}$, and by the definition in equation (23), all $\boldsymbol{L}_n = \boldsymbol{I}$. Similarly, for right-canonical MPS's it is true that all $\boldsymbol{R}_n = \boldsymbol{I}$. As an immediate consequence, $Z = \boldsymbol{L}_N = \boldsymbol{R}_1 = 1$ so that the tensor is automatically a probability tensor.

While the canonicity constraint might look complicated, it is actually easy to describe the set of cores for which it holds true. Defining stacked core matrices,

$$\boldsymbol{U}_n^{(L)} = \begin{pmatrix} \mathcal{G}_n[1] \\ \vdots \\ \mathcal{G}_n[K_n] \end{pmatrix}, \quad \boldsymbol{U}_n^{(R)} = \begin{pmatrix} \mathcal{G}_n[1]^\top \\ \vdots \\ \mathcal{G}_n[K_n]^\top \end{pmatrix} \tag{33}$$

the constraints are simply ${\boldsymbol{U}_n^{(L)}}^\top \boldsymbol{U}_n^{(L)} = I$ and ${\boldsymbol{U}_n^{(R)}}^\top \boldsymbol{U}_n^{(R)} = I$ for left- and right-canonical MPS's, respectively. By inspection, the constraints are equivalent to the stacked matrices being column-orthogonal. Recalling the algorithm used to construct tensor trains using sequential SVD's from section 2.2, this condition is natural: There we defined the cores as reshapes of orthogonal matrices, and as it turns out, $\boldsymbol{U}_n = \boldsymbol{P}\boldsymbol{U}_n^{(L)}$ for a permutation $\boldsymbol{P}$, so that the cores returned by the algorithm define a left-canonical MPS. The final step of the algorithm returns a scalar coefficient, scaling the normalized tensor train appropriately.

Strict left/right-canonical form is only possible if we impose some constraints on the ranks r_n. For left-canonical matrices, $\boldsymbol{U}_n^{(L)}$ has shape $K_n r_{n-1} \times r_n$, so we need $r_n \leq K_n r_{n-1}$ to hold to ensure that the matrix can be column-orthogonal. If we assume $r_n = K_n r_{n-1}$ and $K = K_n$ for all n, this leads to the following progression of ranks

$$1, K, K^2 \ldots K^2, K, 1. \tag{34}$$

Since the maximal rank grows exponentially, this is further evidence that we need to impose some additional low-rank assumption.

While canonicity is a computationally beneficial constraint to impose, it is not enough to ensure uniqueness. For a left/right-canonical MPS, we can still preserve both canonicity and the value of the MPS if we substitute $\mathcal{G}_n[i_n]$ for $\boldsymbol{Q}_n^\top \mathcal{G}_n[i_n] \boldsymbol{Q}_{n+1}$ for orthogonal matrices $\boldsymbol{Q}_n$. Schollwöck (2011) gives a good introduction to a unique representation alternately called the Vidal representation or the $\Gamma\Lambda$-representation as it defines (Vidal, 2003),

$$\mathcal{G}_n[i_n] = \boldsymbol{\Lambda}_{n-1} \boldsymbol{\Gamma}_n[i_n], \tag{35}$$

where $\boldsymbol{\Lambda}_n$ is a diagonal matrix and $\boldsymbol{\Gamma}_n$ is the same size as $\mathcal{G}_n[i_n]$. We additionally impose the constraint that $\mathcal{G}_n[i_n]$ is left-canonical,

$$\boldsymbol{I} = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]^\top \mathcal{G}_n[i_n] = \sum_{i_n=1}^{K_n} \boldsymbol{\Gamma}_n[i_n]^\top \boldsymbol{\Lambda}_{n-1}^2 \boldsymbol{\Gamma}_n[i_n]. \tag{36}$$

There are of course many such possible decompositions. To make the representation unique, we note that,

$$\mathcal{T}_{\mathcal{I}} = \ldots \boldsymbol{\Lambda}_{n-1} \underbrace{(\boldsymbol{\Gamma}_n[i_n] \boldsymbol{\Lambda}_n)}_{\tilde{\mathcal{G}}_n} \boldsymbol{\Gamma}_{n+1}[i_{n+1}] \ldots, \tag{37}$$

where $\tilde{\mathcal{G}}_n$ describes an equivalent set of cores, defining the same tensor. It can then be shown that we can simultaneously require that this alternative representation corresponds to a right-canonical set of cores (Vidal, 2003; Schollwöck, 2011),

$$\boldsymbol{I} = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]\mathcal{G}_n[i_n]^\top = \sum_{i_n=1}^{K_n} \boldsymbol{\Gamma}_n[i_n]\boldsymbol{\Lambda}_n^2\boldsymbol{\Gamma}_n[i_n]^\top. \tag{38}$$

This is a particularly powerful representation as it allows us to shift effortlessly between left- and right-canonical core sets by simply collecting the $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ cores in different orders. In particular, we can employ mixed-canonical representations, where $\boldsymbol{L}_{n'} = \boldsymbol{I}$ for all $n' \leq n$ and $\boldsymbol{R}_{n'} = \boldsymbol{I}$ for all $n' > n$. A univariate marginal can then be computed in constant time as

$$p(x_n = i_n) = \mathrm{Tr}[\mathcal{G}_n[i_n]^\top \boldsymbol{L}_{n-1}\mathcal{G}_n[i_n]\boldsymbol{R}_{n+1}] = \mathrm{Tr}[\boldsymbol{\Gamma}_n[i_n]^\top \boldsymbol{\Lambda}_{n-1}^2\boldsymbol{\Gamma}_n[i_n]\boldsymbol{\Lambda}_n^2]. \tag{39}$$

2.5 Sampling

An attractive feature in a probabilistic model is the ability to generate random samples. A direct approach to generate samples is to use an ancestral sampling routine based on the marginals and conditionals we have already discussed (Ferris and Vidal, 2012; Han et al., 2017). We will sample sequentially from the left, starting with x_1, then x_2, and so on. x_1 is simple to sample, as the marginal is readily available as

$$p(x_1 = k) = \mathrm{Tr}[\mathcal{G}_1[k]^\top \mathcal{G}_1[k]\boldsymbol{R}_2]. \tag{40}$$

Having sampled x_1, we can then go on to sample $x_2|x_1$ from the appropriate conditional distribution,

$$p(x_2 = k|x_1) \propto \mathrm{Tr}[\mathcal{G}_2[k]^\top \left(\mathcal{G}_1[x_1]^\top \mathcal{G}_1[x_1]\right) \mathcal{G}_2[k]\boldsymbol{R}_3], \tag{41}$$

and so on,

$$p(x_n = k|x_1, \dots x_{n-1}) \propto \mathrm{Tr}[\mathcal{G}_n[k]^\top \boldsymbol{C}_n\mathcal{G}_n[k]\boldsymbol{R}_{n+1}], \tag{42}$$

where the conditioning information of the past samples is summarized in a matrix

$$\boldsymbol{C}_n = \mathcal{G}_{n-1}[x_{n-1}]^\top \dots \mathcal{G}_1[x_1]^\top \mathcal{G}_1[x_1] \dots \mathcal{G}_{n-1}[x_{n-1}]. \tag{43}$$

This sampling routine offers an excellent argument for why canonical forms can be useful: If we pick our MPS to be right-canonical, we can remove all of the right marginal matrices $\boldsymbol{R}_n = \boldsymbol{I}$ from the equations above, reducing the computational load.

3. Inference for the MPS

We consider a joint distribution $p(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{Y}$ is observed and $\boldsymbol{X} = \{X_n\}_{n=1}^N$ is a set of unobserved discrete variables, such that the joint distribution can be studied as a unnormalized probability tensor indexed by $\boldsymbol{X}$. Calculating the posterior distribution $p(\boldsymbol{X}|\boldsymbol{Y})$ by Bayes' theorem

$$p(\boldsymbol{X}|\boldsymbol{Y}) = \frac{p(\boldsymbol{X}, \boldsymbol{Y})}{p(\boldsymbol{Y})} \tag{44}$$

reduces to finding the evidence $p(\boldsymbol{Y})$ which is the normalization constant. If the domain of $\boldsymbol{X}$ is of moderate size, this is easily computed as the sum across all possible configurations, but this calculation suffers under a combinatorial explosion as N grows large.

We can instead consider approximations based on a finite number of observations of the probability tensor. As the tensor is unnormalized, the approximation problem is ill-posed. Consider, for instance, the scenario where all elements but one have been evaluated; if the final element is vanishingly small, the approximation is excellent, but if it has a value vastly larger than the observed elements, it can be arbitrarily poor.

Using an MPS as the approximate model alleviates this to a degree, as we limit our search to sufficiently regular probability tensors that can be written as a low-rank MPS. As such we hope to exploit that observing a minority of configurations will still inform us heuristically about the value of similar configurations.

While we could approximate the tensor directly and normalize post hoc, we will advocate using a divergence measure between probability distributions. In particular, we will attempt to use variational inference which minimizes the Kullback-Leibler divergence,

$$\mathrm{KL}(q(\boldsymbol{X})\|p(\boldsymbol{X}|\boldsymbol{Y})) = \mathbb{E}_q\left[\ln \frac{q(\boldsymbol{X})}{p(\boldsymbol{X}|\boldsymbol{Y})}\right], \tag{45}$$

by solving the equivalent problem of maximizing the evidence lower bound (ELBO),

$$\ln p(\boldsymbol{Y}) \geq \mathcal{L} = \mathbb{E}_q\left[\ln \frac{p(\boldsymbol{X},\boldsymbol{Y})}{q(\boldsymbol{X})}\right]. \tag{46}$$

If q is parametric with parameters θ, which we denote q_θ for now, then the most common gradient estimators are (Ranganath et al., 2014; Kingma and Welling, 2014)

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\boldsymbol{X}\sim q}\left[\left(\ln \frac{p(\boldsymbol{X},\boldsymbol{Y})}{q(\boldsymbol{X})}\right)\nabla \ln q_\theta(\boldsymbol{X})\right] = \tag{47}$$
score estimator

$$\mathbb{E}_{\epsilon\sim q_0}\left[\nabla_\theta \ln \frac{p(h(\epsilon,\theta),\boldsymbol{Y})}{q(h(\epsilon,\theta))}\right] \tag{48}$$
reparametrization

The score estimator is alternately known as the black-box gradient estimator or REINFORCE (Ranganath et al., 2014), while the reparametrization estimator depends on finding a function $h(\epsilon,\theta)$ and distribution q_0 such that $\hat{\boldsymbol{X}} = h(\epsilon,\theta)$ for $\epsilon \sim q_0$ is identically distributed to $\boldsymbol{X} \sim q_\theta(\boldsymbol{X})$.

The reparametrization estimator is considered superior as it has empirically lower variance. Constructing a differentiable reparametrization for a discrete distribution is however impossible, but it has recently been shown that the gradient can be approximated with some fidelity using a relaxed differentiable version of the discrete distribution.

3.1 Differentiable MPS

As we saw in section 2.5, ancestral sampling from the MPS involves sampling from categorical distributions. To reparametrize a categorical random variable $k \sim \mathrm{Categorical}(\boldsymbol{p})$ with probability vector $\boldsymbol{p}$, we can employ the Gumbel-max trick which provides a non-differentiable reparametrization (Gumbel, 1954)

$$w_k = p_k + G_k, \quad k = \operatorname*{argmax}_k(w_k), \tag{49}$$

where $G_k \sim \text{Gumbel}(0,1)$ are i.i.d. standard Gumbel random variables. Here, the argmax is the non-differentiable component, and the Gumbel variables take the role of the non-parametric random noise ϵ. To relax this non-differentiable component, we substitute it with

$$\text{onehot}(\underset{k}{\text{argmax}}\, w_k) \approx \text{softmax}(\boldsymbol{w}/T), \tag{50}$$

with the fidelity of the approximation increasing as the temperature $T \to 0$. This approximation is alternately known as Gumbel-softmax (Jang et al., 2017) or the concrete distribution (Maddison et al., 2017).

While this allows us to (approximately) reparametrize every step of the ancestral sampling process, it does not take into account that the categorical distribution of x_n,

$$p(x_n = k|x_1, \dots x_{n-1}) \propto \text{Tr}[\mathcal{G}_n[k]^\top \boldsymbol{C}_n \mathcal{G}_n[k] \boldsymbol{R}_{n+1}], \tag{51}$$

as seen in equation (42), depends conditionally on the exact samples $x_m, m < n$ from the past, via the recursively-defined conditioning tensor,

$$\boldsymbol{C}_n = \mathcal{G}_{n-1}[x_{n-1}]^\top \boldsymbol{C}_{n-1} \mathcal{G}_{n-1}[x_{n-1}]. \tag{52}$$

To get around this, we can define a relaxed conditioning matrix,

$$\hat{\boldsymbol{C}}_n = \sum_{k=1}^{K} \hat{x}_{nk} \mathcal{G}_{n-1}[k]^\top \hat{\boldsymbol{C}}_{n-1} \mathcal{G}_{n-1}[k], \tag{53}$$

which is based on a separate sequence of $\hat{\boldsymbol{x}}_n = \text{onehot}(\hat{x}_n)$ continuous random variables, which we design to be Gumbel-softmax relaxations of the conditional sampling equation in equation (51),

$$z_{nk} = \log \text{Tr}[\mathcal{G}_n[k]^\top \hat{\boldsymbol{C}}_n \mathcal{G}_n[k] \boldsymbol{R}_{n+1}] \tag{54}$$

$$\hat{\boldsymbol{x}}_n = \text{softmax}\left(\frac{1}{T}(\boldsymbol{z}_n + \boldsymbol{G}_n)\right). \tag{55}$$

where we have swapped the proper conditioning matrix $\boldsymbol{C}$ for the relaxed version and where $\boldsymbol{G}_n$ is a vector of stacked i.i.d. Gumbel variables. As the conditioning matrix is a convex combination, we can also impose a mixture interpretation of the approximate conditional,

$$p(\hat{x}_n = k|\hat{x}_1, \dots \hat{x}_{n-1}) \propto \sum_{m=1}^{K} \hat{x}_{nk} \text{Tr}[\mathcal{G}_n[k]^\top \left(\mathcal{G}_{n-1}[m]^\top \hat{\boldsymbol{C}}_{n-1} \mathcal{G}_{n-1}[m]\right) \mathcal{G}_n[k] \boldsymbol{R}_{n+1}], \tag{56}$$

which is a mixture of the different conditionals that would arise based on the sampled value of x_n, weighted by the normalized Gumbel-softmax variables $\hat{x}_n$. By continuing to expand the conditioning matrices, we can get a mixture over all possible pasts $\hat{x_1}, \dots, \hat{x}_{n-1}$.

Note that in the low-temperature limit, this approaches the correct conditional sampling steps. We name the sequential procedure generating the $\hat{x}_n$ variables the differentiable MPS (dMPS), as every step of it is differentiable.

3.2 Unbiased Gradient Estimation

While using the dMPS as a direct substitute for the MPS during inference will often work fine, the consecutive approximations mean that we are no longer solving the original problem, but some facsimile thereof.

To improve this approximation, we can make use of recently proposed efficient unbiased gradient estimators for objectives involving discrete random variables, with the REBAR estimator and its generalization RELAX at the forefront (Tucker et al., 2017; Grathwohl et al., 2017). The RELAX estimator is unbiased and takes the form

$$\nabla_\theta \mathbb{E}_{q(x;\theta)}[f(x)] = \mathbb{E}\big[(f(x) - \mathbb{E}_{q(z|x;\theta)}[c_\phi(z)])\nabla_\theta \ln q(x;\theta)\big] + \tag{57}$$

$$\nabla_\theta \mathbb{E}_{q(z;\theta)}[c_\phi(z)] - \mathbb{E}_{q(z;\theta)}\big[\nabla_\theta \mathbb{E}_{q(z|b;\theta)}[c_\phi(z)]\big] \tag{58}$$

where f is a loss function or other scalar function of interest, c_ϕ is a parametric control variate, $x \sim q(x;\theta)$ is the discrete variable sampled from parametric probabilistic model of interest q, and z is any random variable; ideally one strongly correlated with the original variable x. The estimator typically works best when $q(z;\theta)$ and $q(z|b;\theta)$ can be assumed to be reparametrizable.

In the original papers, z is picked to be continuous reparametrization of x in the sense that $x = H(z)$ for some non-differentiable transfer function $H(\cdot)$. They then recommend setting $c_\phi(z) = f(\tilde{H}(z))$ where $\tilde{H}(z)$ is a differentiable approximation of H. RELAX further advocates augmenting c_ϕ as,

$$c_\phi(z) = f(\tilde{H}(z)) + c'_\phi(z), \tag{59}$$

where c' is a neural network or other high-capacity model (Grathwohl et al., 2017).

To design a RELAX estimator for the MPS, we must find a continuous variable $\boldsymbol{z}_n$ that correlates with each of the $\boldsymbol{x}_n = \text{onehot}(x_n)$ discrete variables in our model. It would be natural if we could use the differentiable relaxation $\hat{\boldsymbol{x}}_n$, but this is not immediately correlated with $\boldsymbol{x}_n$—in fact, the two define independent sequences of random variables. To correlate the two, we take $\boldsymbol{x}_n$ to be generated via a Gumbel-max reparametrization as in equation (49), and we then tie the Gumbel noise used in each true sampling step to that used for the matching relaxed sampling steps in equation (55). The corresponding coupled graphical model is depicted in figure 1. This approach is strongly inspired by one presented in the appendices of the original REBAR paper (Tucker et al., 2017).

Conditioning on $\boldsymbol{x}$, we can sample the Gumbel noise $\boldsymbol{G}_n$ conditioned on that information (see appendix A for details), and using that we can run the coupled dMPS forward. Given the continuous random variables $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}_n\}_{n=1}^N$, we will assume our control variate is,

$$c_\phi(\hat{\boldsymbol{X}}) = \ln p(\hat{\boldsymbol{X}}, \boldsymbol{Y}) - \ln q(\hat{\boldsymbol{X}}) + c'_\phi(\hat{\boldsymbol{X}}), \tag{60}$$

This is not strictly correct as p and q are defined over discrete domains, but often there is a direct way to relax both. For an MPS q, we can relax it by doing a weighted marginalization,

$$\hat{\boldsymbol{L}}_1 = \sum_{i_1=1}^{K_1} \hat{x}_{1i_1} \mathcal{G}_1[i_1]^\top \mathcal{G}_1[i_1], \quad \boldsymbol{L}_n = \sum_{i_n=1}^{K_n} \hat{x}_{ni_n} \mathcal{G}_n[i_n]^\top \hat{\boldsymbol{L}}_{n-1} \mathcal{G}_n[i_n]. \tag{61}$$

This corresponds to standard indexing when the $\hat{\boldsymbol{x}}_n$ vectors are one-hot.

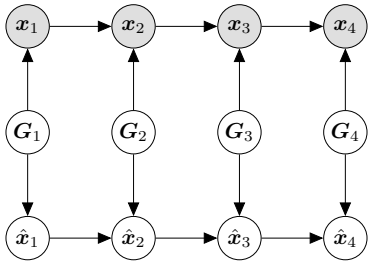


Figure 1: Coupled sampler for RELAX reparametrization. We can sample the continuous variables $\hat{\boldsymbol{x}}_i$ conditioned on the discrete $\boldsymbol{x}_i$.

3.3 Differentiable Normal Forms

Following the previous chapter, we can apply gradient-based optimization to the cores $\mathcal{G}_n[k]$, but we are left with some challenges, such as how to handle normalization. If we constrain the parameters to left/right-canonical cores, we not only get automatic normalization, but also reduce the issue of non-identifiability. We saw in section 2.4 that the defining constraint of left/right-canonical cores was equivalently stated in terms of an orthogonal matrix as in equation (33). By inverting this argument, we can take any column-orthogonal matrix and then define the cores (or their transposes, for right-canonicity) of an MPS to be equal to the blocks of the auxiliary orthogonal matrix. The respective canonical constraint will then be automatically maintained.

Given this, we need a differentiable parametrization of orthogonal matrices. There are in general two overall strategies for constructing orthogonal matrices: i) map a latent quantity through a map that has the orthogonal matrices as its image, or ii) construct a general orthogonal matrix by multiplying members of some set of elementary matrices (Shepard et al., 2015). The drawback of the former approach is that it tends to involve complex matrix operations like matrix inverses and matrix exponentials, while the latter results in long chains of repeated operations.

The Cayley transform (Shepard et al., 2015) says that

$$\boldsymbol{Q} = (\boldsymbol{I} - \boldsymbol{A})(\boldsymbol{I} + \boldsymbol{A})^{-1} \tag{62}$$

is orthogonal, when $\boldsymbol{A}$ is skew-symmetric, i.e., $\boldsymbol{A}^\top = -\boldsymbol{A}$. $\boldsymbol{I} + \boldsymbol{A}$ is always invertible, and is often well-conditioned as it has eigenvalues $1 + i\lambda_k$, where $i\lambda_k$ are the eigenvalues of the skew-symmetric matrix.

Alternatively, to avoid the matrix inverse, we can express an arbitrary orthogonal matrix $\boldsymbol{Q} \in O(k)$ as the product of K Householder reflections (Sun and Bischof, 1995),

$$\boldsymbol{Q} = \prod_{k=1}^{K} H(\boldsymbol{v}_k), \quad H(\boldsymbol{v}_k) = \boldsymbol{I} - 2\frac{\boldsymbol{v}_k \boldsymbol{v}_k^\top}{\boldsymbol{v}_k^\top \boldsymbol{v}_k}, \quad \boldsymbol{v}_k \neq \boldsymbol{0} \tag{63}$$

The complexity of the representation is $\mathcal{O}(K^3)$, so equal to the Cayley transform, but does not involve matrix inverses. On the other hand, the algebraic inverses can also lead to numerical issues, although we empirically observe that this is not common.

4. Symmetry

Clustering and mixture models are often described in terms of a random label x_n determining which cluster or mixture component each data point belongs to. As such, the individual label spaces of all the variables x_n can be identified with the same canonical label space of $K \equiv K_1 = \ldots = K_N$ discrete labels.

Typically, the labels are indistinguishable in the prior leading to both the prior and the posterior possessing a *relabeling symmetry*: given any permutation map $\sigma(\cdot)$ on the K labels, the configuration given by $\hat{x}_n = \sigma(x_n)$ is exactly as probable as the original configuration x_n (so $p(\boldsymbol{x}) = p(\hat{\boldsymbol{x}})$ for distribution p). In other words, for any particular partition of the data, the common label assigned to the elements in each set is inconsequential, as long as it encodes the same partition.

4.1 Marginals under Relabeling Symmetry

Mathematically, we can write distributions with relabeling symmetry as

$$p(\boldsymbol{x}) = \sum_{S \in \mathcal{S}_K(\mathcal{I})} \omega_S \frac{\chi(\boldsymbol{x} \in \mathcal{L}_K(S))}{K!}, \tag{64}$$

where $\sum_S \omega_S = 1$, $\mathcal{S}_K(\mathcal{I})$ is the set of partitions of the set $\mathcal{I}$ into K non-empty parts, where each partition element S is a set of disjoint subsets of $\mathcal{I}$, with their union being $\mathcal{I}$. $\mathcal{L}_K(S)$ is the set of $K!$ labellings $\boldsymbol{x}$ of the partition S using K labels.

Relabeling symmetry has strong effects on the marginals of the distribution, as they by definition are uniform. Stating the factorization formula

$$p(\boldsymbol{x}|S) = \sum_{k=1}^{K} \frac{\chi(x_j = k, j \in A)}{K} \frac{\chi(\boldsymbol{x}_{\mathcal{I}/A} \in \mathcal{L}_{K-1}(S/\{A\}))}{(K-1)!} \chi(x_j \neq k, j \notin A) \tag{65}$$

for any $A \in S$, we can consider $A = N(n, S)$ where $N(n, S)$ is the set of indices grouped together with index n under partition S. Then we can compute

$$p(x_n = k) = \sum_{S \in \mathcal{S}_K(\mathcal{I})} \omega_S \sum_{\boldsymbol{x}_{/n}} p(\boldsymbol{x}|S) = \sum_{S \in \mathcal{S}_K(\mathcal{I})} \omega_S \frac{\chi(x_j = k, j \in N(n, S))}{K} = \frac{1}{K}, \tag{66}$$

where the second equality is true as we marginalize exactly over $p(\boldsymbol{x}_{\mathcal{I}/A}|S/\{A\})$ for $K-1$ labels, and the final equality is true by design.

Using the same type of arguments, we can find that the probability table of the bivariate marginal can be written as a constant plus diagonal matrix

$$p(x_n = k, x_m = \ell) = \begin{cases} \alpha_{nm}, & k = \ell \\ \frac{1 - K\alpha_{nm}}{K^2 - K}, & k \neq \ell \end{cases} \tag{67}$$

where $K\alpha_{nm}$ is the co-occurrence (co-clustering) probability that $x_n = x_m$. In other words, the distribution only distinguishes between whether the two points in question are clustered together, or apart.

4.2 Tensor Trains and Relabeling

In tensor language, we can say that a probability tensor has relabeling symmetry if its underlying distribution has it, and the relabeling symmetry means that the tensor is unchanged under a simultaneous and identical permutation along each mode, i.e.

$$\mathcal{T} = \mathcal{T} \times_1 \boldsymbol{P}_\sigma \times_2 \ldots \times_N \boldsymbol{P}_\sigma \tag{68}$$

has to hold for each permutation σ, where $\boldsymbol{P}_\sigma$ is the matrix representation of σ, where $P_{ij} = 1$ if $\sigma(j) = i$, and zero otherwise. Adopting a general result from Kolda and Bader (2009), we can vectorize to get the equivalent matrix expression

$$\left(\boldsymbol{P}_\sigma^{\otimes N} - \boldsymbol{I}\right) \mathrm{vec}(\mathcal{T}) = 0 \tag{69}$$

where we use $\boldsymbol{P}_\sigma^{\otimes N} = \overbrace{\boldsymbol{P}_\sigma \otimes \ldots \otimes \boldsymbol{P}_\sigma}^{\text{N times}}$ to denote a Kronecker power. So the vectorized tensor has to live in the intersection of the null spaces of all of these massive $K^N \times K^N$ matrices. This statement can be made a bit more compact by noting that the set of all permutations can be constructed from a smaller subset of generating permutations. A classical choice is the set of all transpositions $i \leftrightarrow j$, swapping elements i and j, while the most compact consists of just two elements: a transposition of the two first elements $0 \leftrightarrow 1$, and the cycle permutation $\sigma(i) = i + 1 \pmod K$ that performs a circular shift of every index (Conrad; Miller, 1901).

Permutations on the coordinates of a tensor in a TT format are straightforward to apply, giving rise to a permuted tensor which can also be represented in TT format, with the cores being a simple reordering of the original cores: $\hat{\mathcal{G}}_n[k] = \mathcal{G}_n[\sigma(k)]$. It is of interest to consider whether we can encode knowledge about the posterior, such as the relabeling symmetry, directly into the MPS approximation as this might reduce both redundancy in representation and the risk of degenerate solutions.

In all generality, there is a trivial construction for making a tensor (or other function-like object) invariant to a finite closed group of transformations $\sigma \in \Pi$, which is to construct a new tensor by averaging over all the group elements

$$\hat{\mathcal{T}}_{\mathcal{I}} = \sum_{\sigma \in \Pi} \mathcal{T}_{\sigma(\mathcal{I})}. \tag{70}$$

This approach has been used previously in the kernel literature to make invariant kernels (Haasdonk and Burkhardt, 2007), and is simple to apply in the TT setting as well, as we have rules for adding tensor trains together. The problem with this approach arises when the cardinality $|\Pi|$ is large, as the rank grows linearly with the number of terms in the sum, and many groups of interest, including the set of permutations, possess a large number of elements. As such it is more attractive if we can find representations that directly encode the symmetries.

Limiting ourselves to $K = 2$, it is fairly simple to build tensor trains with symmetries using structured cores. This case, while minimally complex in some sense, is of interest due to its tight relationship to the concepts of bits and bit-strings. Huckle et al. (2013) consider a number of bit-string symmetries, including bit-shift, reverse, and relabeling, the latter of

which they name the bit-flip symmetry. For $K = 2$, it is characterized by the single swap permutation $0 \leftrightarrow 1$. They further show that if

$$\mathcal{G}_n[1] = \boldsymbol{U}_n \mathcal{G}_n[0] \boldsymbol{U}_{n+1} \tag{71}$$

with $\boldsymbol{U}$ being a set of involutions where $\boldsymbol{U}_n^2 = \boldsymbol{I}$ and $\boldsymbol{U}_{N+1} = \boldsymbol{U}_0 = 1$, we can inspect the value of the tensor at index $\mathcal{I}$

$$\mathcal{T}_{\mathcal{I}} = \ldots (U_n \mathcal{G}_n[i_n] U_{n+1})(U_{n+1} \mathcal{G}_{n+1}[i_{n+1}] U_{n+2}) \ldots = \tag{72}$$

$$\ldots \mathcal{G}_n[\bar{i}_n] \mathcal{G}_{n+1}[\bar{i}_{n+1}] \ldots = \mathcal{T}_{\bar{\mathcal{I}}} \tag{73}$$

to see that the value at $\mathcal{I}$ is equal to that at index $\bar{\mathcal{I}}$ where the bar indicates the application of the bit-flip/swap permutation. The authors go on to show that for any tensor train with relabeling symmetry, there exists a representation where the cores follow the above relation (Huckle et al., 2013, Theorem 3.8). Note that the result is not air-tight: cores of the form above lead to the symmetry, and for every tensor train with the symmetry, there exists a set of cores with the properties above, but this does not directly exclude different sets of cores describing the same tensor, but without the property. Indeed, in the proof of theorem 3.8, they have to double the assumed rank to prove that the tensor train has a set of cores obeying equation (71). In the next section we will pursue a more general theory of symmetry. Our approach is inspired by the physics literature, where they have developed MPS's invariant to various symmetries, in particular of the continuous and spatial variety (Singh et al., 2010, 2011; Weichselbaum, 2012).

4.3 Representation Theory

In section 2.4, we noted that each tensor does not have a unique representation as a tensor train, and that a family of gauge transforms,

$$\mathcal{G}_n[k] \to \boldsymbol{A}_n^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1}, \tag{74}$$

leave the implicit tensor invariant, as the $\boldsymbol{A}_n$ matrices cancel. If we conjecture that this is a necessary condition, such that the equivalence of two equal-sized tensor trains implies that they are gauge transforms of each other, we get the consequence that for tensor trains invariant to permutation, the permutation must be acting as a gauge transform (Bridgeman and Chubb, 2017). In other words,

$$\mathcal{G}_n[\sigma(k)] = \boldsymbol{A}_{n,\sigma}^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1,\sigma}. \tag{75}$$

We note that $\boldsymbol{A}_{n,\sigma}$ cannot depend on k, by the definition of the gauge transform. We can then also consider consecutive applications of multiple transforms, e.g.,

$$\mathcal{G}_n[\sigma_2(\sigma_1(k))] = \boldsymbol{A}_{n,\sigma_2}^{-1} \boldsymbol{A}_{n,\sigma_1}^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1,\sigma_1} \boldsymbol{A}_{n+1,\sigma_2} = \boldsymbol{A}_{n,\sigma_2 \circ \sigma_1}^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1,\sigma_2 \circ \sigma_1} \tag{76}$$

The last equality states that the matrix representation for the composite map $\sigma_2 \circ \sigma_1$ should be equivalent to the product of the separate matrix representations for σ_1 and σ_2. If we assume that the set of transforms we are interested in form a finite group G under function composition, then the set of matrices obeying this relationship are exactly the

matrix representations of said group. As permutations form groups, we can apply the above idea to our cases of interest.

Any finite group is isomorphic to a subgroup of the symmetric group $S(K)$ by Cayley's theorem. Every element $\sigma \in S(K)$ is a permutation on K elements, and we can define the so-called permutation representation $\rho : S(K) \to \mathbb{R}^{K\times K}$, consisting of the permutation matrices $\rho(\sigma) = \boldsymbol{P}_\sigma$. Specifically, this assumes a correspondence between element k and canonical basis vector $\boldsymbol{e}_k$, so that $\boldsymbol{P}_\sigma \boldsymbol{e}_k = \boldsymbol{e}_{\sigma(k)}$ (where we use $\sigma(k)$ to mean application of the permutation operation). Similarly, we can define another equivalent matrix representation by changing the basis as $\tilde{\boldsymbol{P}}_\sigma = \boldsymbol{Q}\boldsymbol{P}_\sigma\boldsymbol{Q}^{-1}$. We can furthermore construct matrix representations of size $nK \times nK$ by using the direct sum $\oplus$ as

$$(\boldsymbol{Q}_1\boldsymbol{P}_\sigma^{(1)}\boldsymbol{Q}_1^{-1}) \oplus (\boldsymbol{Q}_2\boldsymbol{P}_\sigma^{(2)}\boldsymbol{Q}_2^{-1}) = \begin{pmatrix}\boldsymbol{Q}_1 & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{Q}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{P}_\sigma^{(1)} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{P}_\sigma^{(2)}\end{pmatrix}\begin{pmatrix}\boldsymbol{Q}_1 & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{Q}_2\end{pmatrix}^{-1}, \tag{77}$$

where the resulting matrix is likewise a representation. While we have maintained an explicit basis above, note that this disappears under the gauge transform equivalence, so we will assume that $\boldsymbol{Q} = \boldsymbol{I}$ going forward.

If we assume that our maximum TT rank is a multiple of K, all of the intermediate ranks r_n will be multiples of K as well, and we can use direct sums of the standard permutations matrices to form representations $\boldsymbol{A}_{n,\sigma} = \boldsymbol{I} \otimes \boldsymbol{P}_\sigma$ at every site. Note that their inverse is given by their transpose.

To finalize the representation, we can select K elements $\sigma_k \in S(K)$ where $\sigma_k(1) = k$, to generate the cores

$$\mathcal{G}_n[k] = (\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k})\mathcal{G}_n[1](\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k})^\top, \tag{78}$$

implying that the representation is determined by the choice of matrix representation and a single "free" $\mathcal{G}_n[1]$. From equation (75), a constraint on $\mathcal{G}_n[1]$ is imposed, as it should be invariant to σ where $\sigma(1) = 1$. For the standard permutation representation presented above, the invariant subspace under this subgroup of permutations is the span of $\boldsymbol{e}_1$, and $\boldsymbol{1}$ which is invariant to all permutations. To ensure that the block-wise column- and row-space is the subspace spanned by these vectors, we can write

$$\mathcal{G}_n[1] = (\boldsymbol{I} \otimes \boldsymbol{V})\boldsymbol{B}_n(\boldsymbol{I} \otimes \boldsymbol{V})^\top, \quad \boldsymbol{V} = \begin{pmatrix}\boldsymbol{e}_1 & \frac{1}{\sqrt{K}}\boldsymbol{1}\end{pmatrix}, \quad \boldsymbol{B}_n \in \mathbb{R}^{\frac{2}{K}r_n \times \frac{2}{K}r_{n+1}}. \tag{79}$$

Alternatively, we can use a projection matrix to the same effect, but the above representation is less redundant.

4.4 Left-Canonical Form for the Relabeling Symmetric MPS

Due to the design, we can construe the parameter matrix as a block matrix of 2×2 blocks, $\boldsymbol{B}_n = \sum_{i,j} \boldsymbol{e}_i\boldsymbol{e}_j^\top \otimes \boldsymbol{B}_{nij}$, and use that to consider the inner product

$$\mathcal{G}_n[k]^\top\mathcal{G}_n[k] = (\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k}\boldsymbol{V})\boldsymbol{B}_n^\top(\boldsymbol{I} \otimes \boldsymbol{\Omega})\boldsymbol{B}_n(\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k}\boldsymbol{V})^\top = \tag{80}$$

$$\sum_{ijrs}(\boldsymbol{e}_j\boldsymbol{e}_i^\top\boldsymbol{e}_r\boldsymbol{e}_s^\top \otimes \boldsymbol{V}_k\boldsymbol{B}_{nij}^\top\boldsymbol{\Omega}\boldsymbol{B}_{nrs}\boldsymbol{V}_k^\top) = \tag{81}$$

$$\sum_{js}\boldsymbol{e}_j\boldsymbol{e}_s^\top \otimes \boldsymbol{V}_k\left(\sum_{r=1}^{r_n/K}\boldsymbol{B}_{nrj}^\top\boldsymbol{\Omega}\boldsymbol{B}_{nrs}\right)\boldsymbol{V}_k^\top, \tag{82}$$

where $\boldsymbol{V}_k$ is the same as $\boldsymbol{V}$, with $\boldsymbol{e}_1$ swapped for $\boldsymbol{e}_k$ and $\boldsymbol{\Omega} = \boldsymbol{V}^\top \boldsymbol{V}$. If we standardize with respect to $\boldsymbol{\Omega}$ by setting $\tilde{\boldsymbol{B}}_n \leftarrow (\boldsymbol{I} \otimes \boldsymbol{\Omega}^{1/2})\boldsymbol{B}_n$, we can write the left-canonicity condition from section 2.4 as

$$\boldsymbol{I} = \sum_{k=1}^{K} \mathcal{G}_n[k]^\top \mathcal{G}_n[k] = \sum_{js} \boldsymbol{e}_j \boldsymbol{e}_s^\top \otimes \sum_{k=1}^{K} \boldsymbol{V}_k \left(\sum_{r=1}^{r_n/K} \tilde{\boldsymbol{B}}_{nrj}^\top \tilde{\boldsymbol{B}}_{nrs} \right) \boldsymbol{V}_k^\top. \tag{83}$$

This condition translates into the block-wise statement that for $j = s$, the right Kronecker factor should be the identity matrix, and for $j \neq \ell$ it should be the zero matrix.

Let $\boldsymbol{M} = \sum_{r=1}^{r_n/K} \tilde{\boldsymbol{B}}_{nrj}^\top \tilde{\boldsymbol{B}}_{nrs}$. Splitting the products with $\boldsymbol{V}_k$ into rank-one elements yields

$$\sum_{k=1}^{K} \boldsymbol{V}_k \boldsymbol{M} \boldsymbol{V}_k^\top = \sum_{k=1}^{K} \left(M_{11} \boldsymbol{e}_k \boldsymbol{e}_k^\top + \frac{M_{22}}{K} \mathbf{1}\mathbf{1}^\top + \frac{1}{\sqrt{K}} \left(M_{12} \boldsymbol{e}_k \mathbf{1}^\top + M_{21} \mathbf{1} \boldsymbol{e}_k^\top \right) \right) = \tag{84}$$

$$M_{11} \boldsymbol{I} + \left(M_{22} + \frac{M_{12} + M_{21}}{\sqrt{K}} \right) \mathbf{1}\mathbf{1}^\top, \tag{85}$$

This immediately gives us the condition that $M_{11} = 1$ for diagonal blocks $j = s$ and $M_{22} = 0$ when $j \neq s$. Inspecting $\tilde{\boldsymbol{B}}_n$, this means that all columns with odd index should form an orthonormal basis $\boldsymbol{Q}_{\text{odd}}$. For all blocks it should furthermore hold that

$$M_{22} + \frac{1}{\sqrt{K}} (M_{12} + M_{21}) = 0 \tag{86}$$

which we can express in matrix form as

$$\boldsymbol{Q}_{\text{even}}^\top \boldsymbol{Q}_{\text{even}} + \frac{1}{\sqrt{K}} (\boldsymbol{Q}_{\text{even}}^\top \boldsymbol{Q}_{\text{odd}} + \boldsymbol{Q}_{\text{odd}}^\top \boldsymbol{Q}_{\text{even}}) = \boldsymbol{0} \Rightarrow \tag{87}$$

$$\boldsymbol{H}^\top \boldsymbol{H} + \frac{1}{\sqrt{K}} \boldsymbol{H}^\top + \frac{1}{\sqrt{K}} \boldsymbol{H} + \bar{\boldsymbol{H}}^\top \bar{\boldsymbol{H}} = \boldsymbol{0}, \tag{88}$$

where we get the last equation by defining $\boldsymbol{Q}_{\text{even}} = \boldsymbol{Q}_{\text{odd}} \boldsymbol{H} + \bar{\boldsymbol{Q}}_{\text{odd}} \bar{\boldsymbol{H}}$, where $\bar{\boldsymbol{Q}}_{\text{odd}}$ is the orthogonal subspace of $\boldsymbol{Q}_{\text{odd}}$, and $\boldsymbol{H}$ and $\bar{\boldsymbol{H}}$ are appropriately shaped coefficient matrices. By completing the square, we can transform the condition into

$$(\sqrt{K}\boldsymbol{H} + \boldsymbol{I})^\top (\sqrt{K}\boldsymbol{H} + \boldsymbol{I}) = \boldsymbol{I} - K \bar{\boldsymbol{H}}^\top \bar{\boldsymbol{H}}. \tag{89}$$

Taking the SVD of $\bar{\boldsymbol{H}} = \boldsymbol{L}\boldsymbol{S}\boldsymbol{W}^\top$ we can multiply by $\boldsymbol{W}^\top$ on both sides to get a diagonal matrix on the right-hand side,

$$\boldsymbol{W}^\top (\sqrt{K}\boldsymbol{H} + \boldsymbol{I})^\top (\sqrt{K}\boldsymbol{H} + \boldsymbol{I}) \boldsymbol{W} = \boldsymbol{I} - K\boldsymbol{S}^2. \tag{90}$$

Equation (89) has an inner product on the left-hand side, so equation (89) is implicitly solved when

$$\boldsymbol{U} \equiv (\sqrt{K}\boldsymbol{H} + \boldsymbol{I}) \boldsymbol{W} (\boldsymbol{I} - K\boldsymbol{S}^2)^{-\frac{1}{2}}, \tag{91}$$

is an orthogonal matrix. Via algebra, a family of solutions can be found as

$$\boldsymbol{H} = \frac{1}{\sqrt{K}} \left(\boldsymbol{U} (\boldsymbol{I} - K\boldsymbol{S}^2)^{\frac{1}{2}} \boldsymbol{W}^\top + \begin{pmatrix} \boldsymbol{U} & \bar{\boldsymbol{U}} \end{pmatrix} \begin{pmatrix} \boldsymbol{C}_1 \\ \boldsymbol{C}_2 \end{pmatrix} \bar{\boldsymbol{W}}^\top - \boldsymbol{I} \right), \tag{92}$$

where for $\boldsymbol{U}$ and $\boldsymbol{W}$, the matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{W}}$ are orthogonal to their counterparts, and span the orthogonal complement of bases $\boldsymbol{U}$ and $\boldsymbol{W}$, respectively. If we use $\boldsymbol{U}_*$ and $\boldsymbol{W}_*$ to denote the concatenation of the two complementary bases into a full basis (square orthogonal matrix),

$$\boldsymbol{H} = \frac{1}{\sqrt{K}} \left(\boldsymbol{U}_* \boldsymbol{C}_* \boldsymbol{W}_*^\top - \boldsymbol{I} \right), \quad \bar{\boldsymbol{H}} = \boldsymbol{L}\boldsymbol{S}\boldsymbol{W}^T, \quad \boldsymbol{C}_* = \begin{pmatrix} (\boldsymbol{I} - K\boldsymbol{S})^{\frac{1}{2}} & \boldsymbol{C}_1 \\ \boldsymbol{0} & \boldsymbol{C}_2 \end{pmatrix} \tag{93}$$

fulfills the projected condition from equation (90) for arbitrary orthogonal $\boldsymbol{U}$, $\boldsymbol{W}$, $\boldsymbol{L}$ and diagonal $\boldsymbol{S}$ with $|S_{ii}| < \frac{1}{K}$. Plugging the result back into the original constraint from equation (89), we get the final condition

$$\boldsymbol{I} = \boldsymbol{C}_*^\top \boldsymbol{C}_* + \begin{pmatrix} K\boldsymbol{S}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & (\boldsymbol{I} - K\boldsymbol{S}^2)^{\frac{1}{2}} \boldsymbol{C}_1 \\ \boldsymbol{C}_1^\top (\boldsymbol{I} - K\boldsymbol{S}^2)^{\frac{1}{2}} & \boldsymbol{C}_1^\top \boldsymbol{C}_1 + \boldsymbol{C}_2^\top \boldsymbol{C}_2 \end{pmatrix} \tag{94}$$

which reduces to two simplifying conditions: $\boldsymbol{C}_1 = \boldsymbol{0}$ and $\boldsymbol{C}_2$ has to be orthogonal.

For $\boldsymbol{S} = \boldsymbol{0}$, there is no weight on the orthogonal subspace, and

$$\boldsymbol{H} = \frac{1}{\sqrt{K}} \left(\boldsymbol{U}\boldsymbol{W}^\top - \boldsymbol{I} \right), \tag{95}$$

where $\boldsymbol{Q}\boldsymbol{W}^\top$ is an arbitrary orthogonal matrix.

Just like with the original left/right-canonical forms, we cannot expect the above to work with arbitrary choices of rank: the identity matrix has full-rank, and by adding K rank-r matrices we can get at best a rank Kr matrix. Problematically, the above design is intrinsically of lower rank than it should be: the coefficient matrix $\boldsymbol{B}_n$ is only rank $\min(\frac{2}{K} r_n, \frac{2}{K} r_{n+1})$. Multiplying by K, we see that for the initial regime where $r_n < r_{n+1}$, we have to ensure $2r_n \geq r_{n+1}$. Maximizing the possible rank by setting $2r_n = r_{n+1}$, we get the rank progression

$$1, K, 2K, 2^2K, \ldots, 2^nK, \ldots K, 1. \tag{96}$$

Since this maximal rank is smaller than the maximal rank of the unconstrained problem, it might be the case that this is not simply due to the relabeling symmetry constraints, but that there exists symmetric tensors that are not expressible using the above representation. Indeed, this appears to be the case empirically. A simple remedy is to augment the size of the core to the maximal rank achievable in the unconstrained setting by zero padding the relabeling symmetric representation, and then add the same constant matrix to all core slices. While this appears to remove the low-rank problem, we have been unable to find a parametric canonical form for this augmented representation, which is left as future work.

5. Relationship to Probabilistic and Graphical Models

Whereas we have transformed the probabilistic model into a probability tensor and then approximated it with the MPS in the sense of a tensor approximation, it is possible to do it the other way around. In fact, there is a duality between tensor networks (the generalization of MPS's to more complex hierarchies of tensor products) and probabilistic graphical models (Robeva and Seigal, 2018). This dual view lends additional insights: For example,

an efficient ordering of tensor contractions when computing e.g. a marginal or the normalization constant can be found by using the classical junction-tree algorithm on the dual graphical model (Robeva and Seigal, 2018). Prior to the explicit description of the duality, it was also shown how graphical models (or the corresponding decomposed log-densities) could be mapped to tensor networks (Novikov et al., 2014).

These two results highlight that we can in some sense characterize the MPS as a graphical model for which inference is particularly efficient. We find that MPS's bear resemblance to observable operator models (Jaeger, 2000) and the class of rational stochastic languages (Balle et al., 2015) which are formally distributions over all sequences Σ^* using some alphabet of states Σ where the distribution of $\boldsymbol{x} = (i_1, \ldots, i_N) \in \Sigma^*$ is computable as,

$$p_{\mathrm{RSL}}(\boldsymbol{x}) = \boldsymbol{v}_0^\top \boldsymbol{A}_{i_1} \ldots \boldsymbol{A}_{i_N} \boldsymbol{v}_\infty \tag{97}$$

for some set of matrices $\boldsymbol{A}_k$. The MPS is tremendously similar, except that it is a distribution over sequences of length N only and allows for the state matrices $\boldsymbol{A}$, including dimensionality, to vary with the index. If we truncate the model above to only be over length N sequences, we can identify it with an MPS with the translation invariance property, where $\mathcal{G}_m[k] = \mathcal{G}_n[k]$ for all m, n (Schollwöck, 2011). This similarity is of interest, as the most prominent member of the above model classes is the Hidden Markov model (HMM), which can be converted into the above form by setting

$$\boldsymbol{A}_i = \mathrm{Diag}(\boldsymbol{O}_{s,:})\boldsymbol{T} \tag{98}$$

for transitions $T_{ij} = p(z_{n+1} = j | z_n = i)$, latent states $z_n \in \Sigma$, and emission matrix $O_{ij} = p(x_n = j | z_n = i)$ where $\boldsymbol{O}_{s,:}$ denotes the s'th row (Jaeger, 2000). We highlight this as the observable operator model is a generalization of the HMM, which is limited by equation (98) constraining the shape of each $\boldsymbol{A}_k$ matrix (Jaeger, 2000). For instance, an HMM can never have negative elements in its matrices.

6. Experiments

In this section we will demonstrate the functionality of the matrix product state model when used as a variational approximation for tensor-shaped distributions. Throughout, we will use a stochastic block model (Nowicki and Snijders, 2001) posterior as our approximation target. It is a model for community detection in networks, which is related to other clustering problems. In particular, we assume an observed binary adjacency matrix $\boldsymbol{Y} \in \{0, 1\}^{N \times N}$ for an undirected graph over N vertices. The stochastic block model for K communities is

then,

$$\boldsymbol{w} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K), \tag{99}$$
cluster proportions

$$x_i \sim \text{Categorical}(\boldsymbol{w}), \qquad \forall i \in 1, \ldots, N, \tag{100}$$
cluster assignments

$$\eta_{k\ell} \sim \text{Beta}(a_{k\ell}, b_{k\ell}), \qquad \forall k \in 1, \ldots, K, \ell \in k, \ldots, K, \tag{101}$$
link probabilities

$$\boldsymbol{Y}_{ij} \sim \text{Bernoulli}(\eta_{x_i, x_j}), \qquad \forall i \in 1, \ldots, N, j \in i+1, \ldots, N. \tag{102}$$
observed links

Here, $\eta_{k,\ell}$ describes the probability of a connection between community k and ℓ, with $\ell \geq k$, and $\{\alpha_k\}_{k=1}^K$, $\{a_{k\ell}\}_{k,\ell=1}^K$, and $\{b_{k\ell}\}_{k,\ell=1}^K$ are hyperparameters. We consider the label symmetric setting $\alpha_k = \alpha, a_{k\ell} = a, b_{k\ell} = b \forall k, \ell \in 1, \ldots, K$. This model cannot be written as a probability tensor in its current form, as it contains continuous variables $\boldsymbol{w}$ and $\eta_{k\ell}$. Fortunately, the model is in the conjugate exponential family, so we can integrate out $\boldsymbol{w}$ and $\eta_{k\ell}$, leaving a posterior density over the discrete x_i.

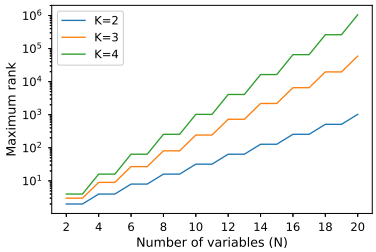
$$P(\boldsymbol{X}|\boldsymbol{Y}) \propto B(\alpha_1 + n_1, \ldots, \alpha_K + n_k) \prod_{k\ell} B(m_{k\ell} + a, \bar{m}_{k\ell} + b), \tag{103}$$

where B is the (multivariate) Beta function, n_k denotes the number of observations in cluster k and $m_{k\ell}$, and $\bar{m}_{k\ell}$ denotes the number of edges and non-edges respectively between nodes in cluster k and ℓ. If we let each x_i index a dimension of a tensor, we get a $K \times \ldots \times K$ (unnormalized) posterior probability tensor with N modes. This tensor will be our approximation target.

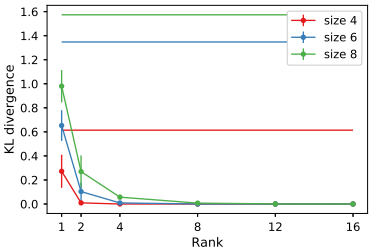
6.1 Influence of Rank

The rank of the matrix product state is the most significant tuning parameter, both in terms of modeling capacity and computational complexity. Using the incremental SVD algorithm defined in section 2.2, we are guaranteed to always be able to perfectly approximate any tensor, if only we use a sufficiently high rank. In the worst case, each iteration involves the SVD of a matrix with maximal rank, i.e. a matrix of size $K^i \times K^{N-i}$ in the i'th step, until we hit step $\lfloor N/2 \rfloor$, at which point the effective rank will start to decrease. The maximal rank is attained at this tipping point, so the maximal rank required to express a tensor with N modes of length K is $K^{\lfloor N/2 \rfloor}$. We plot this in figure 2a for different values of K. This is smaller than the number of elements in the tensor, but unfortunately scales in the same exponential manner. As such, low-rank assumptions are a necessity.

To give a brief demonstration of the model's potential efficacy, we generated small Erdős-Rényi random graphs with 4, 6 and 8 vertices. These are sufficiently small for us to compute the true gradient and true posterior, making it possible to calculate the actual KL divergence between the posterior and the approximation. We found the locally optimal approximation using an off-the-shelf BFGS optimization routine and 10 random restarts, and selected the solution with the smalles KL divergence. We did this for 10 random



(a) The upper bound on the rank of the true TT decomposition of any tensor with N modes of dimensionality K.



(b) The true KL divergence computed for small tractable models and approximations of varying rank. Errorbars denote standard deviation over 10 random graphs. Horizontal lines denote best mean-field solution.

instances for each vertex count. Figure 2b shows the average KL divergence at the best run, with the errorbars denoting the standard deviation across different random graphs. While the tensors in question here are very small with only 16, 64 and 256 elements respectively, the maximal ranks are 4, 8, and 16, but in this experiment the approximation already appears quite trustworthy at around half that rank. Note that this result might depend a lot on how the random graph is generated, especially for larger graphs.

For further demonstration, we fitted models of varying rank using BFGS (as above) and unfolded the resulting tensors into a matrix. Naive unfoldings of a tensor have a tendency to hide its structural regularity, in much the same way as a random permutation on e.g. an adjacency graph can make even very regular graphs appear irregular. To form a structured unfolding, we can group the random variables (for graph clustering, one for each vertex) into two equally-sized ordered lists, which we then to use to index into the rows and columns of the matrix, respectively. We associate the i'th element x_i of the first list with the i'th digit of a base K number, i.e. $r = \sum_{i=0}^{N/2-1} x_i K^i$, which we can then use as a row index; we can do the same for the second list and the column index. In this way, we can map any configuration of the discrete random variables to an element in the matrix. We call the resulting matrix the mosaic of the tensor. We show mosaics of differently ranked tensor

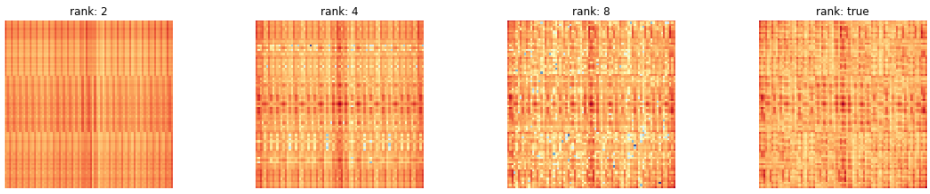
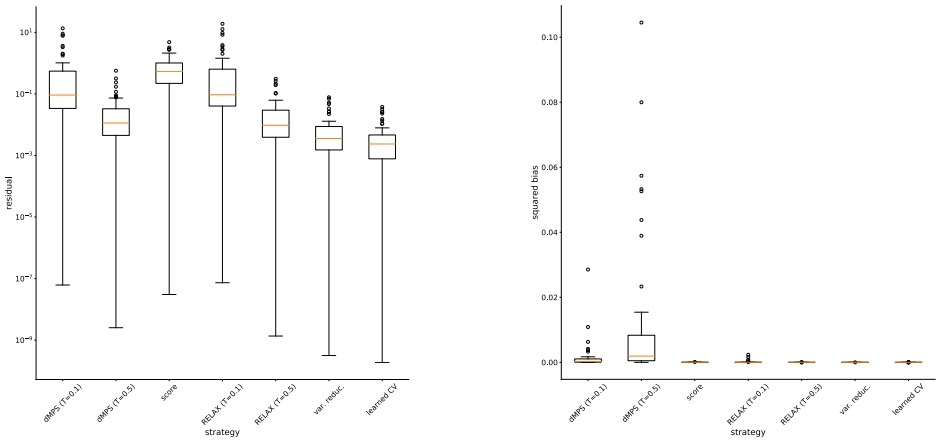


Figure 3: Mosaics of varying rank, showing a tiled heatmap of the posterior approximation. Structural complexity increases with rank, slowly approximating the complexity of the true tensor.



(a) Variance of the gradient estimators.

(b) Most of our methods are unbiased, except for inference using the dMPS directly.

Figure 4: Estimated squared bias and variance to true gradient with 20 000 gradient samples, each with 100 gradient draws per estimate. We register the estimated bias and variance for each parameter—the boxplots summarize the distribution of the estimates over the parameters. We use T to denote temperature used in Gumbel-softmax.

approximations in figure 3, compared to the true posterior tensor in the setting where we have $N = 8$ vertices and $K = 3$ communities.

6.2 Variance Reduction

We have detailed a number of ways to estimate the gradients and control their variance. Keeping within the small N paradigm where the true gradient is tractable, we will demonstrate that the gradients exhibit different characteristics even at this scale. In particular, we take the graph to be an $N = 4$ cycle graph and look for $K = 2$ communities. We initialized randomly, and took 20 BFGS steps to get into the basin of attraction. Empirically, we observed that the gradients at random initializations were be extremely unstable, but we observed that the gradient algorithms tend to escape this regime quickly.

Figure 4 shows the variance and squared bias of various estimators and control variates, including results for two temperatures of the Gumbel softmax. We took a 100-sample Monte Carlo gradient estimator as our gradient estimator, and resampled the estimator 20 000 times to produce the statistical summaries. We used an MPS with a canonical core using the Householder implementation which has 52 parameters. We calculated the summaries for each partial gradient individually, letting the box plots describe the distribution of the quantities across parameters. By using an unbiased estimate of the gradient variance along an unbiased estimate of the squared residual with respect to the true gradient, we can get an unbiased estimate of the squared bias by using the bias-variance trade-off identity.

The score estimator, our baseline, stands out as the highest variance estimator. The dMPS estimators employ a direct differentiation of a dMPS and are the simplest estimators in our arsenal. We note that for both temperature settings we achieve a reduction in variance, but the reduction is particularly significant for $T = 0.5$ which is in the same league as the best estimators. This is offset by its significant bias, and there we see the inverse relationship: the lower temperature estimator has significantly smaller bias. So the dMPS estimators seem to exhibit a very explicit bias-variance trade-off.

Comparing estimators of equal temperature to each other, the two static RELAX estimators (without optimization over the variance-reducing parameters) match the dMPS estimators in terms of variance, but as with all other RELAX-based estimators we see that they are unbiased. To be clear, these estimators use the dMPS as control-variate, without any added parametric component, which goes some way in explaining the correlation with that approach.

The models labeled var. reduc. performs stochastic optimization on the variance by using the variance gradient estimator described in the REBAR and RELAX papers (Tucker et al., 2017; Grathwohl et al., 2017). We optimize for 40 000 steps using the AMSgrad algorithm which has improved convergence properties compared to Adam (Reddi et al., 2018). The most advanced estimator we investigate is the learned CV model where we extend RELAX by using the control variate,

$$\nu(\ln p(\boldsymbol{x}, \boldsymbol{z}) - \ln q_{\text{MPS}}(\boldsymbol{z}) + \alpha \hat{f}(\boldsymbol{z})) \tag{104}$$

where the two first terms make up the dMPS-based control variate, while $\hat{f}$ is a flexible function (a neural network) with everything scaled by a coefficient ν (Grathwohl et al., 2017). Due to the structure of our problem, we settle for using an MPS with canonical cores of maximum rank 2 instead, scaled by α. This works quite well, yielding the best unbiased estimator of the whole selection. It might be possible to improve upon this further by using a higher-rank MPS in the control variate, but there are certainly some complexity trade-offs.

A few empirical observations about training the variance reduced estimators are worth mentioning. That RELAX is better with high-temperature estimators goes against the intuition that we should strive for a control variate that is as close as possible to the true objective; apparently the increased variance we incur from a low temperature is too high a price. In fact, when we optimize for the temperature, we often see it increasing to values of 1 or higher, which is significantly above what we would otherwise expect based on model-fitting intuitions and the literature on Gumbel-softmax where temperatures around 0.5 are usually recommended (Maddison et al., 2017; Jang et al., 2017). The scalar weight ν multiplied onto the control variate is on the other hand quite stable, and almost always converges to 1. The few times ν diverged, it was usually preceded by the temperature dropping to a very low value, causing high variance gradients and unstable training. In general, we advocate keeping both the temperature and the scaling parameter from dropping too close to 0.

6.3 Biased vs Unbiased Gradients

The earliest of the recent bout of papers on gradient-based learning of discrete distributions hinged mostly on relaxations such as the Gumbel-softmax trick we employed to form the

dMPS, which naturally introduced bias into the inference (Maddison et al., 2017; Jang et al., 2017). Although they had some success in their applications, we will make the case that the bias can be quite harmful.

In particular, we ran a dMPS versus an MPS trained with RELAX gradients and a learned control variate on an $N = 9$ graph (specifically the first 9 nodes of Zachary's karate network) taking $K = 2$. We used canonical cores, a gradient estimator based on 1 000 samples, and optimized using AMSgrad. We tracked the true ELBO, which is tractable due to the size of the graph, as well as the differentiable and stochastically estimated version of the ELBO targeted by the dMPS. Both of these learning curves appear in figure 5. Note that similar behavior was noted in several experimental runs, with this one picked for illustrative purposes.

We notice in particular that while the dMPS appears to perform exceedingly well according to its own biased metric, its behaviour with respect to the true objective is concerning, as it dips to its lowest point around iteration 500, before rebounding and leveling off at a higher level, around iteration 800. Most importantly, this seems to indicate that the biased objective is running counter to the true objective in some subtle way.

We can pick this observation apart by taking the entire 512 element posterior probability tensor, and compare it to the approximate posteriors. We plot the true probability against the approximation for all tensor elements individually in the plots in figure 6. The two first subfigures correspond to the iterations marked with vertical lines in figure 5, and the third subfigure is at convergence. We note that at iteration 500 the approximation is somewhat sound, with the approximation roughly in the right range of values and a good fit to the highest value mode. Then at iteration 800, we see that pretty much all of the probability mass is concentrated at the single highest mode. The unbiased gradient meanwhile finds a reasonable approximation, especially with respect to the more significant high probability states.

6.4 Experiments on the karate graph

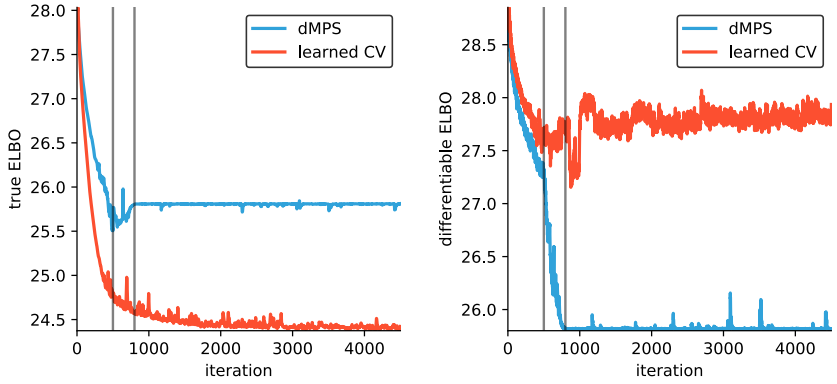
In this section we approximate the posterior of a stochastic block model applied to the venerable karate graph (Zachary, 1977). This is a 34 vertex social graph with $K = 2$ gold standard communities. We use uniform priors, $\alpha_k = 1/K$, $a_{k\ell} = 1$ and $b_{k\ell} = 1$ since informed priors can have a large impact on the posterior's concentration.

The undirected graph has a total of $\binom{34}{2} = 561$ possible edges, and we assume that we only observe 400 of these vertex pairings, leaving a test set of 161 to allow us to evaluate the model by its prediction on held-out data.

As a baseline, we also solved it using a mean-field approximation, employing KL-corrected bounds (Hensman et al., 2012) and an out of the box L-BFGS optimization to efficiently find local optima. We ran it 500 times to establish the global optimum with some certainty.

6.4.1 Initialization and Warm Starts

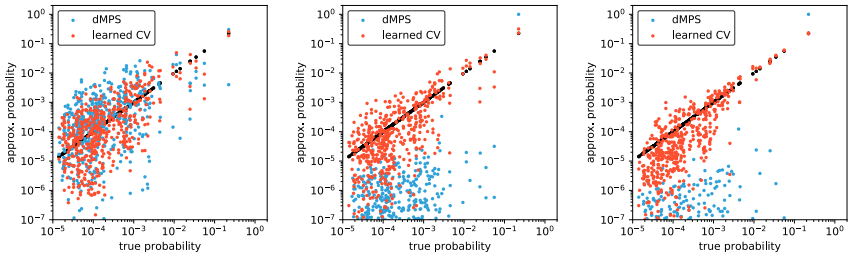
A central issue with graphs of any meaningful size is that due to the combinatorial explosion of possible states, they can be quite sensitive to initialization. One advantage of our



(a) The true ELBO (not stochastically estimated).

(b) The stochastic loss function employed by the dMPS.

Figure 5: Learning curves of two MPS models trained using biased and unbiased gradients, respectively. We report the true loss, and the implicit loss of the biased method. Horizontal lines correspond to snapshots at iteration 500 and 800 in figure 6.



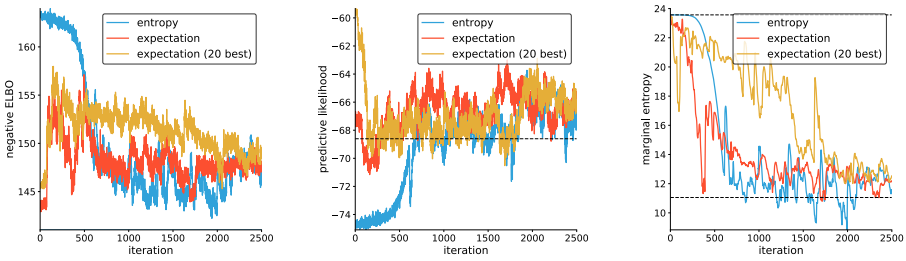
(a) Iteration 500.

(b) Iteration 800.

(c) At convergence.

Figure 6: Difference between the estimated q and the true posterior at all 512 positions of the probability tensor. Black points correspond to the true posterior.

problem is that we can in many cases easily find local optima or efficiently computable approximations, in particular in the settings where we can calculate the analytical ELBO for a mean-field model. To each mean-field solution $q_i(\boldsymbol{X}) = \prod_{n=1}^{N} q_{i,n}(\boldsymbol{x}_n)$, we can associate a tensor $\mathcal{T}_i$ which is the outer product of the marginal probability vectors, as in equation (3).



(a) ELBO for the different initialization strategies.

(b) Predictive likelihood on test set. Line denotes the mean-field model with best ELBO.

(c) Marginal entropy. Lines denote maximum possible entropy (top) and entropy where each marginal puts 0.9 on a single entry.

Figure 7: Learning curves for three different initialization strategies.

We could cobble together all the mean-field solutions into a mixture $\bar{\mathcal{T}} = \frac{1}{S}\sum_{i=1}^{S}\mathcal{T}_i$, and compute the norm

$$\left\langle \mathcal{T} - \bar{\mathcal{T}}, \mathcal{T} - \bar{\mathcal{T}} \right\rangle = \langle \mathcal{T}, \mathcal{T} \rangle + \frac{1}{S^2}\sum_{i,j=1}^{S}\langle \mathcal{T}_i, \mathcal{T}_j \rangle - \frac{2}{S}\sum_{i=1}^{S}\langle \mathcal{T}, \mathcal{T}_i \rangle , \tag{105}$$

which is computable as all of the inner products are simple expectations per equation (18). Unfortunately, due to the high dimensionality of the tensors, this problem appears to be numerically problematic as distances break down due to the curse of dimensionality.

As a light proxy, we propose using the sum of log expectations,

$$\sum_{i=1}^{S}\log\langle \mathcal{T}, \mathcal{T}_i \rangle . \tag{106}$$

Intuitively, if $\mathcal{T}_i$ is close to a one-hot encoding, maximizing the expectation $\langle \mathcal{T}, \mathcal{T}_i \rangle$ encourages the model to put as much mass as possible on that index. Summing over the expectations with respect to all states aims to then find a compromise that puts mass on all the indices, but if the expectations are just summed the optimal solution will be to put maximum mass on the index with highest weight. The logarithm tries to balance this by making it disproportionately disadvantageous to put zero mass on any of the states. This target is amenable to off-the-shelf optimizers like BFGS. Since there is still some chance of collapsing onto the S states, we recommend performing early stopping after 30 steps.

Another alternative to random initialization is to start in the maximum entropy state. The entropy of the entire MPS is difficult to calculate, but since we can compute the marginals efficiently we can use the marginal entropy as a proxy, encouraging that no label preferred a priori.

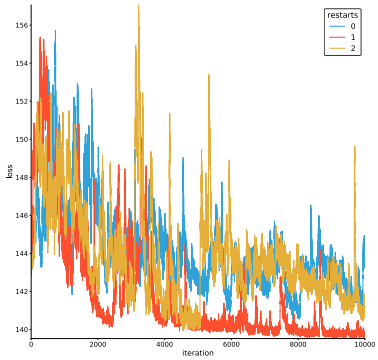


Figure 8: 3 restarts of rank 16 MPS on the karate network. We continue working with the red curve.

In figure 7 we visualize the learning curves generated by initializing using these two strategies. We also consider a boosted version of the expectation-based initialization, where we only maximize the objective with respect to the 20 entries with highest predictive likelihood.

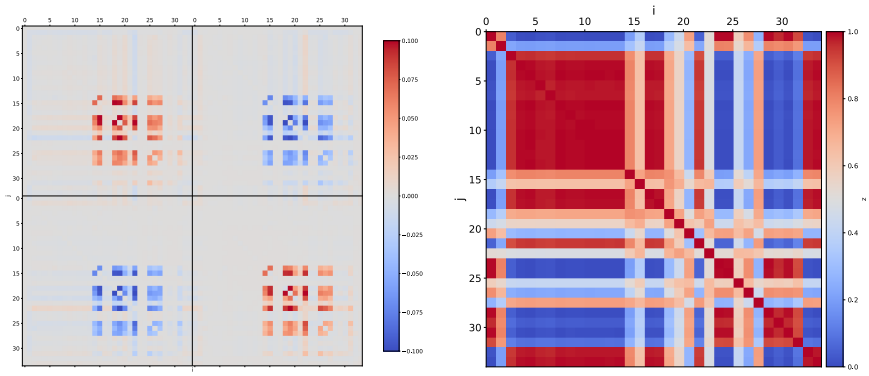
In figure 7a we have inverted the ELBO to get a minimization problem, and we see that all curves implement a decreasing behavior as expected, although the expectation strategy behaves oddly in the initial transient phase. The initial state of the entropy initialization is not surprisingly a rather poor initialization with respect to the ELBO, but it seems to decrease to the level of the others gracefully.

The entropy initialization likewise under-performs on predictive likelihood in figure 7b.The best predictive likelihood achieved by any of the mean-field optima is -59 which is achieved by the boosted method, which was specifically designed to be close to that state, but we see that this is not a stable point and the boosted strategy dives down to the same level as the other curves. This is in line with the mean-field solution with the best bound having a rather mediocre predictive performance, so this is mostly indicative of some kind of model mismatch.

From the marginal entropy plot in figure 7c, we see that all of the states start in fairly entropic solutions, despite having very different empirical behavior. We also see that the inference procedure drives all of them towards low entropy configurations, with the lower reference line being the entropy of a mean-field model where every marginal puts 0.9 probability mass on a single entry, and divides the rest evenly. This is discouraging, as we had hoped the flexibility of the MPS would allow us to find solutions with high marginal entropy.

6.4.2 Induced Covariance

Next, we ran a rank 16 MPS with canonical cores, warm starts using the expectation strategy, and AMSgrad with a 0.01 learning rate. We plot three restarts in figure 8. We note that run 1 surpasses the other two runs by a fair margin, so we selected that for further



(a) Covariance matrix for restart 1. Blocks corresponds to labels. Marginal covariance has been deducted.

(b) Co-location matrix for restart 1.

Figure 9: Covariance and co-location matrices for restart 1, estimated using 10 000 samples.

inspection. As a rule, we generally observe quite noisy transient phases during optimization, but eventually it levels out.

We have no good easily computable metric to evaluate the degree to which dependencies have been encoded in the approximate posterior. Something like mutual information would have been ideal, but to get a more local measure we will consider the covariance matrix of each one-hot categorical variable pair $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n'}$, with element (k, k') of the covariance matrix given by,

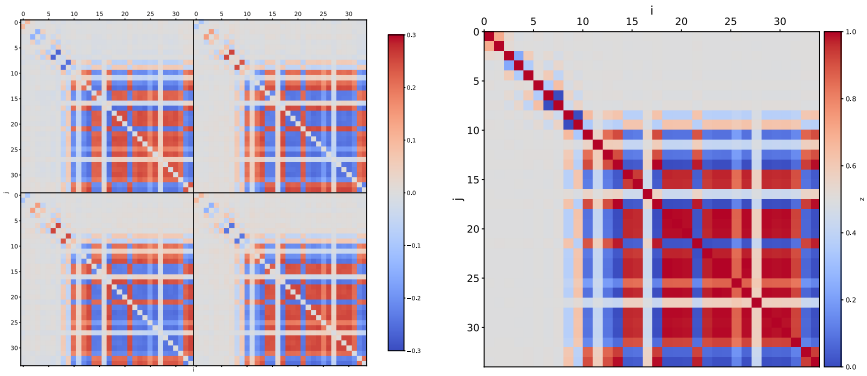
$$\mathrm{Cov}(x_{nk}, x_{n'k'}) = \mathbb{E}[x_{nk}x_{n'k'}] - \mathbb{E}[x_{nk}]\,\mathbb{E}[x_{n'k'}] \tag{107}$$

If $k = k'$, and the respective covariance element is positive, this means that those two points are likely to get sorted into the same cluster. We can take this a step further by computing the co-location matrix

$$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^\top\right] \tag{108}$$

where $[\boldsymbol{X}]_{nk} = x_{nk}$. Given any sample instantiation $\hat{\boldsymbol{X}}$, the quantity $\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^\top$ gives a binary matrix where each element is 1 if two elements share a label in $\hat{\boldsymbol{X}}$. Taking the expectation then gives us the co-location probability of how likely two elements are clustered on average. Both of these quantities can be estimated. The expectations here are computable using the tensor train framework, as we can compute the marginal distribution of the two variables, or use the tower property to express it in terms of a conditional expectation. It is often simpler to just sample it though, which is what we will do going forward, using 10 000 samples.

We visualize the covariance matrix in figure 9a. The blocks are indexed by label, so the upper left block is covariance between x_{i0} and x_{j0}, the upper right block is between elements x_{i0} and x_{j1}, and so on. For saliency, we have deducted the corresponding covariance matrix of the marginal distribution, which is only non-zero on the diagonals of each of these



(a) Covariance matrix for the permutation-invariant core. Blocks corresponds to labels. Marginal covariance has been deducted.

(b) Co-location matrix for the permutation-invariant core.

Figure 10: Covariance and co-location matrices for the permutation-invariant kernel, estimated using 10 000 samples. Note that the color scale for covariance has been extended.

blocks. So all of the observed covariance is due to the higher-rank modeling. We note that one group exhibits relatively high positive correlation, making it likely that these elements group together. When we look to the corresponding co-location matrix in figure 9b, we see that there are some elements that almost always get clustered together, but we also see that the elements for which we observed high covariance are the elements that are somewhat uncertainly labeled. The strong blocks correspond to elements that have highly concentrated marginals, giving them an almost certain assignment. This is a consequence of most of the local optima we find concentrating around a single symmetry mode. The uncertain elements then get randomly assigned to each of these blocks, but the covariance tells us that they are assigned as a group. This is not the case for mean-field models, where all uncertain elements must be randomly assigned without consideration for other elements, which encourages these rank-1 mean-field solutions to collapse unto a specific hard clustering.

6.4.3 Permutation Invariance

We now run a similarly sized MPS with a set of permutation-invariant cores. Seeing as we are working with $K = 2$ we can use the bit-flip symmetry core (Huckle et al., 2013), although we will have to explicitly normalize it as we do not have a canonical self-normalizing version of that core available. After 20 000 iterations, we get the covariance and co-location plots of figure 10.

Contrasted with the canonical core set from before, the covariances exhibited in figure 10a are a lot stronger; note that the color scale has been extended to cover the new range of values. Compared to figure 10b, we also note that the co-location is near-identical to that

of a covariance block, which is of course a direct consequence of the permutation-invariant representation modeling everything through covariance alone. Another consequence is that the diagonal blocks of the covariance matrix are now identical, since the two labels are interchangeable. More distressingly, we note that the upper left corner seems to hardly be modeled at all, which is especially odd as this part of the model was assigned labels with very high certainty. This could be an issue with model capacity, but it could also be a local optima problem: if we want to put the points of the upper left corner in one cluster, it might be problematic if the K clusters have already been "spent" clustering the lower left corner, according to some kind of partition that is not consistent with the remaining points. Either way, it should be clear that permutation-invariant cores have a large effect on the produced posterior approximations and that the choice of core leads to qualitative differences.

7. Related Work and Further Reading

As matrix product states have seen wide-spread use in quantum mechanics, there is a large literature on the topic. Unfortunately, for readers outside the physics community, the presentation can be impenetrable, which is one of the motivating reasons behind the present manuscript. For general introductions to the topic, we recommend Schollwöck (2011) which covers the mechanics and mathematics well and is quite pedagogical. The figure-heavy introduction by Bridgeman and Chubb (2017) complements it nicely. For some more esoteric and physics-oriented introductions, interested readers might want to consult Perez-García et al. (2007); Orús (2014a); Biamonte and Bergholm (2017).

In the tensor literature, the tensor train has also been growing in popularity, starting with its publication in Oseledets (2011). The literature on tensor trains has proven less relevant for this presentation, due to the more algebraic focus. We note that other results from the physics literature have started to percolate over, such as the tensor ring decomposition (Zhao et al., 2016), which is known as a MPS with periodic boundary condition in quantum mechanics (Schollwöck, 2011).

As noted previously, we are not the first to consider the relationship between matrix product states and probabilistic models. Arguably, this relationship is a bit of a false dichotomy, as its application in quantum mechanics means that it has always had a statistical interpretation, by virtue of the way a quantum mechanical wave function is related to a probability distribution by way of Born's rule. The more explicit connections to probabilistic models, and graphical models in particular, were pioneered by Novikov et al. (2014) and culminated recently in a duality result (Robeva and Seigal, 2018). There have also been a few ventures into applying tensor trains and matrix product states in machine learning applications (Stoudenmire and Schwab, 2016; Pestun and Vlassopoulos, 2017). We highlight the recent paper by Han et al. (2017), which uses the probabilistic interpretation of an MPS directly, and propose the use of a sampling algorithm which was originally presented in Ferris and Vidal (2012). Their work differs from ours by only using the model for generative density estimation. By not imposing any prior knowledge or otherwise preventing the density from collapsing unto the observations, we fear this application is somewhat ill-posed.

8. Conclusion

In this paper, we have attempted to lay the groundwork for how matrix product states can be used as an approximate model in variational inference. Part of our contribution is that this text should serve as a pedagogical gateway for people in the machine learning community interested in this topic, by offering an introduction to the topic, relevant references to the existing literature, and translating some of the physics-oriented results into a notation that is more relatable to machine learning researchers.

The main challenge of integrating the MPS into a variational inference setting is finding ways to estimate gradients in a robust manner. As a first step, this requires finding a differentiable representation for the MPS, which is a bit different from other papers where the model is often learned by way of repeated application of singular value decompositions like in the tensor train algorithm we originally described in section 2.2 (Oseledets, 2011), or the density-matrix renormalization group (DMRG) method popular in physics (Schollwöck, 2011). We note that Han et al. (2017) propose a hybrid approach, combining the iterative procedure of the DMRG, gradient steps, and SVD. This is one of the things that could be pursued in future work, although we were worried that local updates might make the model more likely to converge to local modes. This worry is motivated by the passing similarity with Gibbs sampling, which can become trapped due to its inability to make global changes (Jain and Neal, 2004).

There are some remaining mysteries when it comes to representations. We consider a differentiable $\Gamma\Lambda$-representation to be an important target for future research as it would make marginalization operations exceedingly expeditious, and allow direct parametric control over the form of the marginals. Further exploration of symmetry representations is another thing we believe would be fruitful; although modeling symmetry modes wastes some of the model's capacity, we conjecture that factoring in symmetry constraints reduces the search space and thus helps with exploring the model space. One could also attempt to find representations that only concentrate on a single symmetry mode, but we have not looked into this. We found that many of our representations depended on orthogonal matrices with their own differentiable parametric forms, where we have one unresolved issue as the orthogonal matrices do not form a connected manifold, i.e. there does not exist a parametric representation capable of modeling all orthogonal matrices with both positive and negative determinant (Shepard et al., 2015). A final missing piece is the opportunity to use complex valued representations which is standard in the physics community. One might hope that using complex numbers translates directly into added model capacity, simply by virtue of the increase in parameters, without increasing the rank. Additionally, since we only need to model the square root of the true probability tensor, allowing complex values gives us an infinite number of alternative solutions; when the tensor train is real-valued, we can flip signs in the elements of the tensor train without affecting the square. With complex-valued tensor trains we can multiply any element with any root of unity without affecting it. On the other hand, the constructive argument means that it always suffices to use real-valued tensors.

Optimization remains a challenge. While we found that our stochastic gradients performed admirably considering the difficulty of the problem, more work could be put into finding good warm start procedures based on e.g. locally optimal mean-field solutions like

we used. Another avenue would be to extend the coordinate-ascent updates of DMRG and Han et al. (2017) to the full variational problem. Finally, we think there could be merit in pursuing Riemannian optimization as the tensor trains span a sub-manifold of the space of all tensors, and the gauge invariance means that we are often using highly redundant parameterizations. Some progress has already been made in this direction (Steinlechner, 2016). Numerical issues also remain problematic, as we are forced to work in the non-log domain to exploit the MPS structure. Whether this can be fully circumvented by technical means is an open question. One could also imagine hybrid models where we also model the log-probability with an MPS in conjunction with the normalized model.

The MPS methods are designed to scale well, although the $\mathcal{O}(R^3)$ scaling in the maximum rank eventually gets prohibitive. Scaling in N is linear for many operations, which is a big difference from most other factorization schemes. The trade-off is that few of the MPS and TT operations come cheaper than $\mathcal{O}(N)$, e.g. both evaluation and sampling scale as $\mathcal{O}(N)$. We did a full-fledged implementation in Tensorflow, parallelizing where possible, but performance could still be quite slow. Additionally, running it on a GPU often made the whole thing slower, despite most operations being standard linear algebra routines. It is possible that the large computational graphs and the long chains of small matrix operations is a poor fit for Tensorflow. Also, from inspection, the main bottleneck appears to be the existing implementation of Einstein summation, which has not been fully optimized in Tensorflow yet. In summary, it should be possible to make a high-performance versions of the MPS.

There is also room for exploring some of the more advanced architectures from the tensor networks literature. The tensor ring (Zhao et al., 2016), known as an MPS with periodic boundary conditions in the physics literature (Schollwöck, 2011), is a straightforward extension of the tensor train which makes the tensor invariant to cyclic reordering of the cores, but it lacks a canonical representation, as well as the efficient normalization scheme of equation (21), forcing the explicit computation of Kronecker products of the cores. Even more advanced architectures like PEPS and MERA could also prove useful, although few of the analytical formulas available for the MPS carry over (Orús, 2014a,b). Another interesting avenue is the idea of implementing the cores as their own tensor trains, possibly allowing the extension of the MPS ideology to larger ranks (Hübener et al., 2010). Note that tensor trains can also be used to speed up standard matrix multiplications considerably (Oseledets, 2011; Novikov et al., 2015).

References

Borja Balle, Prakash Panangaden, and Doina Precup. A canonical form for weighted automata and applications to approximate minimization. January 2015.

Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, July 2017.

Christopher Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

Jacob C Bridgeman and Christopher T Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and Theo-*

retical, 50(22):223001, May 2017.

Keith Conrad. Generating sets. Technical report, University of Conneticut, Department of Mathematics. URL `www.math.uconn.edu/~kconrad/blurbs/grouptheory/genset.pdf`.

Andrew J Ferris and Guifre Vidal. Perfect sampling with unitary tensor networks. *Physical review. B, Condensed matter*, 85(16):165146, April 2012.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. October 2017.

J E Gumbel. Statistical theory of extreme values and some practical applications, NBS. *Applied Mathematical Series*, 1954.

Bernard Haasdonk and Hans Burkhardt. Invariant kernel functions for pattern analysis and machine learning. *Machine learning*, 68(1):35–61, July 2007.

Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *arXiv preprint arXiv:1709.01662*, September 2017.

James Hensman, Magnus Rattray, and Neil D Lawrence. Fast variational inference in the conjugate exponential family. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2888–2896. Curran Associates, Inc., 2012.

R Hübener, V Nebendahl, and W Dür. Concatenated tensor network states. *New journal of physics*, 12(2):025004, February 2010.

Thomas K Huckle, Konrad Waldherr, and Thomas Schulte-Herbrüggen. Exploiting matrix symmetries and physical symmetries in matrix product states and tensor trains. *Linear and Multilinear Algebra*, 61(1):91–122, January 2013.

Michael C Hughes and Erik Sudderth. Memoized online variational inference for dirichlet process mixture models. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1133–1141. Curran Associates, Inc., 2013.

H Jaeger. Observable operator models for discrete stochastic time series. *Neural computation*, 12(6):1371–1398, June 2000.

Sonia Jain and Radford M Neal. A Split-Merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of computational and graphical statistics: a joint publication of American Statistical Association, Institute of Mathematical Statistics, Interface Foundation of North America*, 13(1):158–182, 2004.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*, April 2017.

Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

T Kolda and B Bader. Tensor decompositions and applications. *SIAM Review*, 51(3): 455–500, 2009.

S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J.R. Statist. Soc. B*, 50(2):157–224, 1988.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

George Abram Miller. On the groups generated by two operators. *Bulletin of the American Mathematical Society*, 7(10):424–426, 1901.

Tom Minka. Divergence measures and message passing. Technical report, Microsoft Research, 2005.

Alexander Novikov, Anton Rodomanov, Anton Osokin, and Dmitry Vetrov. Putting MRFs on a tensor train. In *International Conference on Machine Learning*, pages 811–819. jmlr.org, January 2014.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 442–450. Curran Associates, Inc., 2015.

K. Nowicki and T. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

R Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 2014a.

Román Orús. Advances on tensor network theory: symmetries, fermions, entanglement, and holography. *The European physical journal. B*, 87(11):280, November 2014b.

I Oseledets. Tensor-Train decomposition. *SIAM Journal of Scientific Computing*, 33(5): 2295–2317, January 2011.

David Perez-García, Frank Verstraete, Michael M Wolf, and J Ignacio Cirac. Matrix product state representations. *Quantum Information and Computation*, 7(5-6):401–430, 2007.

Vasily Pestun and Yiannis Vlassopoulos. Tensor network language model. *arXiv preprint arXiv:1710.10248*, October 2017.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, April 2014.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*. sanjivk.com, 2018.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

Elina Robeva and Anna Seigal. Duality of graphical models and tensor networks. *Information and Inference: A Journal of the IMA*, June 2018.

Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of physics*, 326(1):96–192, January 2011.

Ron Shepard, Scott R Brozell, and Gergely Gidofalvi. The representation and parametrization of orthogonal matrices. *The journal of physical chemistry. A*, 119(28):7924–7939, July 2015.

Sukhwinder Singh, Robert N C Pfeifer, and Guifré Vidal. Tensor network decompositions in the presence of a global symmetry. *Physical review. A*, 82(5):050301, November 2010.

Sukhwinder Singh, Robert N C Pfeifer, and Guifre Vidal. Tensor network states and algorithms in the presence of a global U (1) symmetry. *Physical Review B: Condensed Matter and Materials Physics*, 83(11):115125, 2011.

Michael Maximilian Steinlechner. *Riemannian optimization for solving high-dimensional problems with low-rank tensor structure*. PhD thesis, École polytechnique fédérale de Lausanne, 2016.

Edwin Miles Stoudenmire and David J Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807, 2016.

Xiaobai Sun and Christian Bischof. A Basis-Kernel representation of orthogonal matrices. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1184–1196, 1995.

Yee W Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In B Schölkopf, J C Platt, and T Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1353–1360. MIT Press, 2007.

George Tucker, Andriy Mnih, Chris J Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. March 2017.

Charles F Van Loan. Tensor network computations in quantum chemistry. Technical report, 2008.

Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical review letters*, 91(14):147902, October 2003.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, January 2008.

Andreas Weichselbaum. Non-abelian symmetries in tensor networks: a quantum symmetry space approach. February 2012.

Zhiqiang Xu, Yiping Ke, and Yi Wang. A fast inference algorithm for stochastic blockmodel. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 620–629, December 2014.

Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, June 2016.

Appendix A. REBAR for categoricals

The current best method unbiased discrete gradient estimator is the REBAR estimator which constructs a control variate with coupled randomness. If we denote the discrete random variable b and the cost function f, then the target is to estimate $\nabla \mathbb{E}[f(b)]$. Given a source of randomness $p(\epsilon)$ and a continuous reparametrization $z = g(\epsilon, \theta)$ such that $b = H(z)$ after passing through a gating function H, REBAR builds a control variate using a dummy reparametrization $z' \sim p(z|b)$ likely to have high correlation with the original.

If b is a categorical variable $\mathrm{Cat}(\boldsymbol{\alpha})$, a natural reparametrization follows via the Gumbel-max trick. The procedure is simply:

$$\begin{aligned}
\epsilon_i &\sim \mathcal{U}[0,1] \\
z_i &= \ln \alpha_i - \ln(-\ln(\epsilon_i)) \sim \mathrm{Gumbel}(\ln \alpha_i, 1) \\
b &= \arg\max(\boldsymbol{z})
\end{aligned}$$

Going the other way and sampling from $p(\boldsymbol{z}|b)$ is slightly more complicated. It turns out that

$$\begin{aligned}
z_b &\sim \mathrm{Gumbel}\left(\ln \sum_{i=1}^{N} \alpha_i, 1\right) \\
z_i|z_b &\sim, \mathrm{TruncatedGumbel}\left(\ln \alpha_i, 1, z_i \leq z_b\right) \quad \forall i \neq b.
\end{aligned}$$

The Gumbel has pdf and cdf

$$f_G(x; \mu, 1) = \exp(-(x-\mu) - \exp(-(x-\mu))) \tag{109}$$

$$F_G(x; \mu, 1) = \exp(-\exp(-(x-\mu))). \tag{110}$$

so the truncated cdf is conveniently just

$$F_{TG}(x; \mu, 1, x \leq t) = \frac{F_G(x; \mu, 1)}{F_G(t; \mu, 1)}. \tag{111}$$

Finding a reparametrization then just becomes a question of applying the inverse transform sampler of the truncated Gumbel to a uniform variable u, and as

$$F_G^{-1}(p; \mu, 1) = \mu - \ln(-\ln(p)). \tag{112}$$

we have

$$x = F_{TG}^{-1}(u) = F_G^{-1}(F_G(t)u) = \mu - \ln(-\ln F_G(t) - \ln u). \tag{113}$$

If we instead want to reparametrize in terms of a Gumbel variable z, we can use that $u = F_G(z)$ (running the inverse transform sampler in reverse) and then substitute in to get

$$x = \mu - \ln(-\ln F_G(t) - \ln F_G(z)) = \mu - \ln\Big(e^{-(t-\mu)} + e^{-(z-\mu)}\Big). \tag{114}$$

which can finally be reduced to $x = -\ln\big(e^{-t} + e^{-z}\big)$.

Bibliography

Agakov, Felix V and David Barber (November 2004). "An Auxiliary Variational Method". In: *Neural Information Processing*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pages 561–566.

Alvarez, Mauricio A, Lorenzo Rosasco, and Neil D Lawrence (June 2011). "Kernels for Vector-Valued Functions: A Review". In: *Foundations and Trends® in Machine Learning* 4.3.

Amari, Shun-Ichi and Hiroshi Nagaoka (2007). *Methods of Information Geometry*. en. American Mathematical Soc.

Aronszajn, N (1950). "Theory of Reproducing Kernels". In: *Transactions of the American Mathematical Society* 68.3, pages 337–404.

Bamler, Robert et al. (2017). "Perturbative Black Box Variational Inference". In: *arXiv preprint arXiv:1709. 07433*.

Banerjee, Arindam et al. (December 2005). "Clustering with Bregman Divergences". In: *Journal of machine learning research: JMLR* 6, pages 1705–1749.

Bernardo, José M and Adrian F M Smith (2000). *Bayesian theory*. eng. Reprint. Wiley.

Bishop, Christopher M (August 2006). *Pattern Recognition and Machine Learning*. en. New York: Springer.

Bonnevie, R, M N Schmidt, and M Mørup (2017). "Difference-of-Convex optimization for variational kl-corrected inference in dirichlet process mixtures". In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Bottou, Léon (January 2010). "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, pages 177–186.

Bridgeman, Jacob C and Christopher T Chubb (May 2017). "Hand-waving and interpretive dance: an introductory course on tensor networks". en. In: *Journal of Physics A: Mathematical and Theoretical* 50.22, page 223001.

Briol, François-Xavier et al. (December 2015). "Probabilistic Integration: A Role in Statistical Computation?" In: arXiv: `1512.00933 [stat.ML]`.

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (September 2015). "Importance Weighted Autoencoders". In: arXiv: `1509.00519 [cs.LG]`.

Carpenter, Bob et al. (2017). "Stan: A Probabilistic Programming Language". In: *Journal of Statistical Software, Articles* 76.1, pages 1–32.

Chen, Zhe (2003). "Bayesian filtering: From Kalman filters to particle filters, and beyond". In: *Statistics* 182.1, pages 1–69.

Cockayne, Jon et al. (February 2017). "Bayesian Probabilistic Numerical Methods". In: arXiv: `1702.03673 [stat.ME]`.

Damianou, Andreas C (July 2015). "Deep Gaussian Processes and Variational Propagation of Uncertainty". PhD thesis. University of Sheffield.

Damianou, Andreas C and Neil D Lawrence (November 2012). "Deep Gaussian Processes". In: arXiv: `1211.0358 [stat.ML]`.

Evenbly, Glen and Robert N C Pfeifer (October 2013). "Improving the efficiency of variational tensor network algorithms". In: arXiv: `1310.8023 [cond-mat.str-el]`.

Flaxman, Seth et al. (March 2016). "Bayesian Learning of Kernel Embeddings". In: arXiv: `1603.02160 [stat.ML]`.

Gelman, Andrew et al. (2014). *Bayesian Data Analysis*. CRC Press.

Grathwohl, Will et al. (October 2017). "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation". In: arXiv: `1711.00123 [cs.LG]`.

Han, Zhao-Yu et al. (September 2017). "Unsupervised Generative Modeling Using Matrix Product States". In: arXiv: `1709.01662 [cond-mat.stat-mech]`.

Hartikainen, J and S Särkkä (August 2010). "Kalman filtering and smoothing solutions to temporal Gaussian process regression models". In: *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 379–384.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani (2015). "Scalable Variational Gaussian Process Classification". In: *AISTATS*. jmlr.org.

Hensman, James, Magnus Rattray, and Neil D Lawrence (2012). "Fast Variational Inference in the Conjugate Exponential Family". In: *Advances in Neural Information Processing Systems 25*. Edited by F Pereira et al. Curran Associates, Inc., pages 2888–2896.

Hoffman, Matthew D, David M Blei, et al. (May 2013). "Stochastic Variational Inference". In: *Journal of machine learning research: JMLR* 14.1, pages 1303–1347.

Hoffman, Matthew D and Andrew Gelman (2014). "The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo". In: *Journal of machine learning research: JMLR* 15.1, pages 1593–1623.

Huckle, T, K Waldherr, and T Schulte-Herbrueggen (January 2013). "Exploiting Matrix Symmetries and Physical Symmetries in Matrix Product States and Tensor Trains". In: arXiv: 1301.0746 [math-ph].

Jaynes, E T (April 2003). *Probability Theory: The Logic of Science*. en. Cambridge University Press.

Jordan, Michael I et al. (November 1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine learning* 37.2, pages 183–233.

King, Nathaniel J and Neil D Lawrence (January 2006). "Fast Variational Inference for Gaussian Process Models Through KL-Correction". In: *Machine Learning: ECML 2006*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pages 270–281.

Kingma, Diederik P and Max Welling (December 2013). "Auto-Encoding Variational Bayes". In: arXiv: 1312.6114v10 [stat.ML].

Kolda, T and B Bader (2009). "Tensor Decompositions and Applications". In: *SIAM Review* 51.3, pages 455–500.

Koller, Daphne and Nir Friedman (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kom Samo, Yves-Laurent and Stephen Roberts (June 2015). "Generalized Spectral Kernels". In: arXiv: 1506.02236 [stat.ML].

Kom Samo, Yves-Laurent and Stephen J Roberts (October 2015). "p-Markov Gaussian Processes for Scalable and Expressive Online Bayesian Nonparametric Time Series Forecasting". In: arXiv: 1510.02830 [stat.ML].

Kreyszig, Erwin (1978). *Introductory Functional Analysis With Applications*. en. John Wiley & Sons.

Kullback, S and R A Leibler (March 1951). "On Information and Sufficiency". In: *Annals of Mathematical Statistics* 22.1, pages 79–86.

Le, Tuan Anh et al. (2018). "Auto-encoding sequential monte carlo". In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Liseo, Brunero and Nicola Loperfido (February 2003). "A Bayesian interpretation of the multivariate skew-normal distribution". In: *Statistics & probability letters* 61.4, pages 395–401.

Liu, Qiang, Jason D Lee, and Michael I Jordan (February 2016). "A Kernelized Stein Discrepancy for Goodness-of-fit Tests and Model Evaluation". In: arXiv: 1602.03253 [stat.ML].

Maddison, Chris J et al. (May 2017). "Filtering Variational Objectives". In: arXiv: 1705.09279 [cs.LG].

Manton, Jonathan H and Pierre-Olivier Amblard (August 2014). "A Primer on Reproducing Kernel Hilbert Spaces". In: arXiv: 1408.0952 [math.HO].

Minka, Tom (2005). *Divergence measures and message passing*. Technical report. Microsoft Research.

Muandet, Krikamol et al. (May 2016). "Kernel Mean Embedding of Distributions: A Review and Beyond". In: arXiv: 1605.09522 [stat.ML].

Naesseth, Christian et al. (2017). "Reparameterization Gradients through Acceptance-Rejection Sampling Algorithms". In: *Artificial Intelligence and Statistics*. proceedings.mlr.press, pages 489–498.

Nemeth, Christopher et al. (August 2017). "Pseudo-extended Markov chain Monte Carlo". In: arXiv: `1708.05239 [stat.ME]`.

Oates, Chris J, Mark Girolami, and Nicolas Chopin (June 2017). "Control functionals for Monte Carlo integration". In: *Journal of the Royal Statistical Society. Series B, Statistical methodology* 79.3, pages 695–718.

Oseledets, I (January 2011). "Tensor-Train Decomposition". In: *SIAM Journal of Scientific Computing* 33.5, pages 2295–2317.

Perez-Garcia, D et al. (August 2006). "Matrix Product State Representations". In: arXiv: `quant-ph/0608197 [quant-ph]`.

Pestun, Vasily and Yiannis Vlassopoulos (October 2017). "Tensor network language model". In: arXiv: `1710.10248 [cs.CL]`.

Polson, Nicholas G, James G Scott, and Jesse Windle (December 2013). "Bayesian Inference for Logistic Models Using Pólya–Gamma Latent Variables". In: *Journal of the American Statistical Association* 108.504, pages 1339–1349.

Rahimi, Ali and Benjamin Recht (2008). "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems 20*. Edited by J C Platt et al. Curran Associates, Inc., pages 1177–1184.

Rainforth, Tom et al. (February 2018). "Tighter Variational Bounds are Not Necessarily Better". In: arXiv: `1802.04537 [stat.ML]`.

Ranganath, Rajesh, Sean Gerrish, and David M Blei (December 2013). "Black Box Variational Inference". In: arXiv: `1401.0118 [stat.ML]`.

Ranganath, Rajesh, Dustin Tran, and David M Blei (November 2015). "Hierarchical Variational Models". In: arXiv: `1511.02386 [stat.ML]`.

Rasmussen, C E and C K I Williams (2006). *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited.

Rezende, D J, S Mohamed, et al. (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *Proceedings of the*.

Robbins, Herbert and Sutton Monro (September 1951). "A Stochastic Approximation Method". In: *Annals of Mathematical Statistics* 22.3, pages 400–407.

Robeva, Elina and Anna Seigal (October 2017). "Duality of Graphical Models and Tensor Networks". In: arXiv: `1710.01437 [math.ST]`.

Ruiz, Francisco J R, Michalis K Titsias, and David M Blei (2016a). "Overdispersed Black-box Variational Inference". In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. UAI'16. Arlington, Virginia, United States: AUAI Press, pages 647–656.

– (2016b). "The generalized reparameterization gradient". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 460–468.

Salimans, Tim and David A Knowles (December 2013). "Fixed-Form Variational Posterior Approximation through Stochastic Linear Regression". In: *Bayesian analysis* 8.4, pages 837–882.

Sato, Masa-Aki (2001). "Online Model Selection Based on the Variational Bayes". In: *Neural computation* 13.7, pages 1649–1681.

Schein, Aaron, Hanna Wallach, and Mingyuan Zhou (2016). "Poisson-Gamma dynamical systems". In: *Advances in Neural Information Processing Systems 29*. Edited by D D Lee et al. Curran Associates, Inc., pages 5005–5013.

Schollwöck, Ulrich (January 2011). "The density-matrix renormalization group in the age of matrix product states". In: *Annals of physics* 326.1, pages 96–192.

Singh, Sukhwinder, Robert N C Pfeifer, and Guifre Vidal (2011). "Tensor network states and algorithms in the presence of a global U (1) symmetry". In: *Physical Review B: Condensed Matter and Materials Physics* 83.11, page 115125.

Singh, Sukhwinder, Robert N C Pfeifer, and Guifré Vidal (November 2010). "Tensor network decompositions in the presence of a global symmetry". In: *Physical review. A* 82.5, page 050301.

Smola, Alex et al. (2007). "A Hilbert Space Embedding for Distributions". In: *Algorithmic Learning Theory*. Edited by Marcus Hutter, Rocco A Servedio, and Eiji Takimoto. Volume 4754. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pages 13–31.

Speed, T P and H T Kiiveri (March 1986). "Gaussian Markov Distributions over Finite Graphs". en. In: *Annals of statistics* 14.1, pages 138–150.

Sriperumbudur, Bharath K, Kenji Fukumizu, and Gert R G Lanckriet (February 2011). "Universality, Characteristic Kernels and RKHS Embedding of Measures". In: *Journal of machine learning research: JMLR* 12, pages 2389–2410.

Stein, Charles (1972). "A bound for the error in the normal approximation to the distribution of a sum of dependent random variables". In: *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*.

Stoudenmire, Edwin Miles and David J Schwab (2016). "Supervised learning with tensor networks". In: *Advances in Neural Information Processing Systems*, pages 4799–4807.

Teh, Yee W, David Newman, and Max Welling (2007). "A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation". In: *Advances in Neural Information Processing Systems 19*. Edited by B Schölkopf, J C Platt, and T Hoffman. MIT Press, pages 1353–1360.

Titsias, Michalis K (2009). "Variational learning of inducing variables in sparse Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics*, pages 567–574.

Titsias, Michalis K and Miguel Lázaro-Gredilla (2015). "Local Expectation Gradients for Black Box Variational Inference". In: *Advances in Neural Information Processing Systems 28*. Edited by C Cortes et al. Curran Associates, Inc., pages 2620–2628.

Tucker, George et al. (2017). "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models". In: *Advances in Neural Information Processing Systems 30*. Edited by I Guyon et al. Curran Associates, Inc., pages 2627–2636.

Wainwright, Martin J and Michael I Jordan (January 2008). "Graphical Models, Exponential Families, and Variational Inference". In: *Found. Trends Mach. Learn.* 1.1-2, pages 1–305.

Weichselbaum, Andreas (February 2012). "Non-abelian symmetries in tensor networks: a quantum symmetry space approach". In: arXiv: 1202.5664 [cond-mat.str-el].

Williams, Ronald J (May 1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3, pages 229–256.

Wilson, Andrew and Ryan Adams (2013). "Gaussian Process Kernels for Pattern Discovery and Extrapolation". In: *Proceedings of The 30th International Conference on Machine Learning*. jmlr.org, pages 1067–1075.

Winn, John M and Christopher M Bishop (2005). "Variational message passing". In: *Journal of Machine Learning Research*, pages 661–694.

Yedidia, Jonathan S, W T Freeman, and Y Weiss (July 2005). "Constructing free-energy approximations and generalized belief propagation algorithms". In: *IEEE transactions on information theory / Professional Technical Group on Information Theory* 51.7, pages 2282–2312.

Yuille, A L (July 2002). "CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation". In: *Neural computation* 14.7, pages 1691–1722.

Zhao, Qibin et al. (June 2016). "Tensor Ring Decomposition". In: arXiv: 1606.05535 [cs.NA].

Zhu, Zhanxing, Ruosi Wan, and Mingjun Zhong (June 2018). "Neural Control Variates for Variance Reduction". In: arXiv: 1806.00159 [stat.ML].