Generative Modeling and Inference in Directed and Undirected Neural Networks

Patrick Stinson

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

ABSTRACT

Generative Modeling and Inference in Directed and Undirected Neural Networks

Patrick Stinson

Generative modeling and inference are two broad categories in unsupervised learning whose goal is to answer the following questions, respectively: 1. Given a dataset, how do we (either implicitly or explicitly) model the underlying probability distribution from which the data came and draw samples from that distribution? 2. How can we learn an underlying abstract representation of the data? In this dissertation we provide three studies that each in a different way improve upon specific generative modeling and inference techniques. First, we develop a state-of-the-art estimator of a generic probability distribution's partition function, or normalizing constant, during simulated tempering. We then apply our estimator to the specific case of training undirected probabilistic graphical models and find our method able to track log-likelihoods during training at essentially no extra computational cost. We then shift our focus to variational inference in directed probabilistic graphical models (Bayesian networks) for generative modeling and inference. First, we generalize the aggregate prior distribution to decouple the variational and generative models to provide the model with greater flexibility and find improvements in the model's log-likelihood of test data as well as a better latent representation. Finally, we study the variational loss function and argue under a typical architecture the data-dependent term of the gradient decays to zero as the latent space dimensionality increases. We use this result to propose a simple modification to random weight initialization and show in certain models the modification gives rise to substantial improvement in training convergence time. Together, these results improve quantitative performance of popular generative modeling and inference models in addition to furthering our understanding of them.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I first would like to thank my advisor Liam Paninski. When I joined the group, I was amazed by the degree of engagement you had in my projects–it's astounding to me that despite the large volume of research being done in the group you are not only able to keep abreast of the technical details but also contribute a steady stream of original ideas.

I would also like to thank the members of my thesis committee: Niko Kriegeskorte, Larry Abbott, John Cunningham, and Rajesh Ranganath. You were instrumental in supporting a direction for me to take to finish. I would like to thank Niko in particular not only for chairing the committee but also for inspiring me with your far-reaching curiosity, positive attitude, and encouragement.

I would like to thank Leslie Osborne for her confidence in me. It was only after an introduction to information theory and those fly papers that my interest in machine learning started to take off.

I thank Jason Maclean for inspiring me ultimately to take the plunge and attend graduate school. I admired your no nonsense approach to research and the lab environment you cultivated during my time at UChicago. You and your students treated me like one of your own.

I would also like to thank my friends. Dan, for the gym celebrity tiers and Ganon v. Ganon matches. Uygar, for the nights working together in the office. Xuexin, for your enthusiasm for conversation and great generosity with your time. Kenny, for our philosophical conversations that stop time for hours. Erdem, for a friendship that would make Zizek blush. Cat, for countless conversations and schemes.

Finally, I thank Audrey for all of her love, companionship, and support.

To the jokesters who upend our precious silly notions

Chapter 1

Introduction

"We are forced to tell the truth, we are constrained, we are condemned to admit the truth or to discover it." — Michel Foucault

1.1 Overview

Much of what is commonly thought of as cognition can be considered a form of inference; it is usually not the data itself that we care about or even seem to think about but rather a more abstract form of the data. Instead of perceiving the world as streams of 'raw' data from our sensory faculties, we cut through redundancy and nuisance by perceiving on a more general and abstract level: we see objects that belong to various conceptual hierarchies, we hear words and sentences instead of phonemes, etc. Inference takes place at every level of complexity of life, from axonal growth cones inferring directions of chemical gradients from the stochastic pattern of receptors binding to the chemical molecules [Mortimer et al., 2009], to the mind's abstraction of concepts [Tenenbaum et al., 2011], to global inference emergent in message passing across networks of individuals (see e.g., [Xu et al., 2014; Vehtari et al., 2014] for how this could be realized in a computational framework).

Generative modeling, the ability to generate samples from a probability distribution of interest,

is closely tied to inference. In many frameworks, they are learned simultaneously. Intuitively, having a good model of the important abstract elements of the data should make it easier to generate data that contain these elements. In other words, it's easier to draw a chair when you know what a chair is.

Generative modeling provides a means of general-purpose data augmentation whose importance may soon come to the forefront of research as more sophisticated models require more data, but there is a more ambitious goal hidden in the form of a sanity check. If we are able to learn from a dataset to in turn generate data that exhibit characteristics from that dataset, our model may have captured in some form the causal factors that gave rise to the data in our dataset. More succinctly, in the limit that our generative model can produce arbitrarily accurate and diverse samples from a distribution, the model has captured all the information in that distribution.

The contributions in this thesis to generative modeling and inference are as follows: in Chapter 2, we develop a method to estimate partition functions of hard to sample distributions, which we use to improve training of Restricted Boltzmann Machines [Smolensky, 1986]. In Chapters 3 and 4, we shift our focus to variational autoencoders [Kingma and Welling, 2014; D.J. Rezende, 2014]. In Chapter 3, we decouple the inference and generative models by generalizing the prior distribution and improve performance. In Chapter 4, we show that a simple random weight initialization modification can lead to substantial training improvements for VAEs with large latent dimension. We conclude with a discussion in Chapter 5. To improve the flow of the thesis, more dense mathematical work for Chapters 2 and 4 has been moved to Appendices A and B, respectively. We include some negative results in Appendices C and D that the reader can omit without detriment to the rest of the thesis. For the remainder of this chapter, we provide the background necessary for easy reading of these chapters in an effort to make this work self-contained.

1.2 Monte Carlo methods

Monte Carlo methods [Robert and Casella, 2013] comprise a general purpose toolbox for estimating expectations over arbitrary (with mild restrictions) probability distributions. As we will see, essentially all of the quantities we are interested in take the form of expectations, so their utility cannot be overstated.

Suppose we have a function $f(x)$ that we would like to average over some probability distribution $p(x)$. Ignoring mild restrictions on $f(\cdot)$ and $p(\cdot)$, the strong law of large numbers tells us

$$\frac{1}{N}\sum_{n=1}^{N} f(x^{(n)}) \xrightarrow{a.s.} \mathbb{E}_{p(x)}[f(x)], \tag{1.1}$$

where $x^{(n)} \sim p(\cdot)$.

In other words, we can estimate the average of a function of a random variable by simply averaging function evaluations at random samples. Importantly, if we take an expectation over the left hand side of (1.1), we see that our estimate is unbiased; that is, $\mathbb{E}[\widehat{\mathbb{E}[f(x)]}] = \mathbb{E}[f(x)]$.

(1.1) applies so long as we can easily sample from $p(\cdot)$. If $p(\cdot)$ isn't easy to sample from, we can still make use of (1.1) with a little extra work.

One option is rejection sampling. If we pick an easy to sample from distribution $q(x)$ and can bound $p(x)/q(x)$ by $M < \infty$, rejection sampling is done by following Algorithm 1.

The issue with rejection sampling is that M must be picked beforehand and cannot be arbitrarily large since

$$P(\text{accept}) = \mathbb{E}_q\left[\frac{p(x)}{Mq(x)}\right] = \frac{1}{M}.$$

Since M is hard to pick for high dimensional distributions, rejection sampling is usually reserved for low dimensional distributions.

Algorithm 1 Rejection Sampling

initialize $s = 1$

while $s \leq S$ **do**

Sample $x \sim q(\cdot)$ and $u \sim U[0, 1]$.

if $u > \frac{p(x)}{Mq(x)}$ **then**

$x^{(s)} = x$

$s = s + 1$

end if

end while

return $\{x^{(s)}\}_{s=1}^{S}$

A second option is importance sampling, which we can derive by noting

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx \tag{1.2}$$

$$= \int f(x)p(x)\frac{q(x)}{q(x)}dx \tag{1.3}$$

$$= \int f(x)\frac{p(x)}{q(x)}q(x)dx \tag{1.4}$$

$$= \mathbb{E}_q[w(x)f(x)], \tag{1.5}$$

where $q(x)$ is called the importance distribution and $w(x) = \frac{p(x)}{q(x)}$ are the importance weights. If $q(x)$ is easy to sample from, we can just sample from it and weight our evaluations of $f(x)$ with $x \sim q(\cdot)$ by $w(x)$.

A potential problem with importance sampling, however, is if $q(\cdot)$ does not match well with $p(\cdot)$ on some subset of p's support. Intuitively, if there are regions where $q(x)$ is small but $p(x)$ is large, or vice versa, the importance weights will be relatively high or low, respectively. The result is high variance importance weights and consequently a high variance Monte Carlo estimate. As with rejection sampling, high dimensional probability distributions often preclude importance sampling, since the probability of such regions existing is high.

Markov Chain Monte Carlo addresses the issues brought up by rejection and importance

sampling by exploring the space of $x \in \mathcal{X}$ based on a series of jumps. By doing so, we trade off having to find a suitable distribution $q(\cdot)$ to match $p(\cdot)$ globally with introducing temporal correlations between samples.

A Markov Chain is a sequence $x^{(1)}, x^{(2)}, ..., x^{(N)}$ whose dynamics obey the following identity

$$p(x^{(i)}) = T(x^{(i)}|x^{(i-1)}), \tag{1.6}$$

where $T(x^{(i)}|x^{(i-1)})$ is called the transition function. To start the chain, $x_0 \sim p_0(x)$ must be specified.

A distribution $p(x)$ is said to be the invariant distribution of the Markov Chain specified by $T(\cdot|\cdot)$ if

$$\lim_{N\to\infty} P(x^{(N)}|x^{(N)} = x) = p(x). \tag{1.7}$$

That is, we can draw samples whose distribution is arbitrarily close to $p(x)$ by running the Markov Chain for enough steps. For any x_0, it can be shown that if $T(x^{(i)}|x^{(i-1)})$ is irreducible ($T(x^{(i)}|x^{(j)}) > 0$ for all i, j) and aperiodic (it does not get stuck in cycles), the Markov Chain will converge to the invariant distribution. Note in the discrete case, running a Markov Chain is equivalent to a power iteration; consequently, we can conclude that the first eigenvector of T is the invariant distribution $p(x)$ (for continuous distributions, it is the first eigenfunction).

Most of the time though, we are interested in deriving a transition function from a desired invariant distribution–not the other way around. The condition of detailed balance provides a sufficient but not necessary condition for a transition function to have an invariant distribution $p(x)$:

$$p(x^{(i)})T(x^{(i-1)}|x^{(i)}) = p(x^{(i-1)})T(x^{(i)}|x^{(i-1)}). \tag{1.8}$$

The Metropolis-Hastings algorithm [Metropolis et al., 1953] is derived from satisfying detailed balance and contains a free parameter in the form of a proposal distribution. It uses the transition

function

$$T(x^{(i)}|x^{(i-1)}) = A(x^{(i)}|x^{(i-1)})g(x^{(i)}|x^{(i-1)}) + (1 - A(x^{(i)}|x^{(i-1)}))\delta(x^{(i-1)}), \tag{1.9}$$

for some proposal distribution $g(\cdot|\cdot)$ where $A(x^{(i)}|x^{(i-1)}) = \min\left(1, \frac{p(x^{(i)})g(x^{(i-1)}|x^{(i)})}{p(x^{(i-1)})g(x^{(i)}|x^{(i-1)})}\right)$.

Note that multiplying $p(x)$ by a constant does not affect the transition function in (1.9), which enables us to sample from unnormalized distributions.

As a special case, consider the proposal distribution $g(x^{(i)}|x^{(i-1)}) = \delta\left(x_{-j}^{(i)} = x_{-j}^{(i-1)}\right) + p(x_j|x_{-j}^{(i-1)})$, where $x_{-j} \triangleq x \setminus x_j$, called Gibbs sampling [Geman and Geman, 1984]. It is easily shown the probability of accepting a proposal from a conditional distribution of $p(x)$ is 1.

A common choice for the proposal distribution is a symmetric Gaussian $g(x^{(i)}|x^{(i-1)}) = \mathcal{N}(x^{(i)}|0, \Sigma)$. Note, however, that Σ may not be easy to determine and for certain distributions a static transition matrix will lead to poor mixing, the time taken such that a sample from the chain is approximately from the invariant distribution. For example, consider a mixture of two isotropic Gaussians with one component having a large variance and the other having a small variance. If Σ is large, if we are within the flatter component, we will have good mixing, but when we transition to the sharper component, it will be difficult to accept the large jumping proposals, since we are already in a high density region. Conversely, if we have a small Σ, we will be able to move in the sharper component, but movement in the flatter component will be relatively slow.

As the target distribution becomes more difficult to sample from, these simpler methods will no longer be sufficient. We will see examples of more sophisticated methods later, including simulated tempering [Geyer and Thompson, 1995] and Hamiltonian Monte Carlo [Neal, 2011].

1.3 Latent variable modeling

If we believe our data can be described in terms of more abstract but less complicated factors, instead of directly modeling how the data behave, we can instead model how the so-called latent variables behave and how the latent variables influence the data. The utility of such modeling is twofold: we build a probabilistic model of the data, but given a data point, we can also infer (or approximate) the distribution of the underlying latent variables that gave rise to the data point.

Factor analysis [Cattell, 1952] models the data as the sum of linear projections from latent factors. The simplest case is linear Gaussian factor analysis, in which

$$p(z) = \mathcal{N}(z|0, I) \tag{1.10}$$

$$p(x|z) = \mathcal{N}(x|\mu + \Lambda z, \Sigma), \tag{1.11}$$

where x represents our data and z represents our latent variables. To fit this model to the data, we can perform Expectation Maximization (EM) [Dempster et al., 1977].

This simple mixing model has substantial limitations. In particular, the additivity of latent factors restricts the expressiveness of the model, causing Gaussian mixture models to tend to oversmooth. If we deviate much from the Gaussian mixture model, though, we can no longer do EM, as we can no longer evaluate the posterior distribution, since it requires marginalization over the latents:

$$p(z|x) = \frac{p(x, z)}{p(x)} = \frac{p(x, z)}{\int p(x, z)dz}, \tag{1.12}$$

which cannot be evaluated in many (non-conjugate) models.

We will explore two alternatives: 1. using a generalization of EM called variational inference [Jordan et al., 1999] to train models in which the posterior cannot be evaluated 2. using an energy-based model in which inference is easy but sampling is hard.

1.4 Variational inference

Suppose we take our factor analysis model in the previous section and after taking an affine transformation of the factors, we compose the transformation with a nonlinear function $f(\cdot)$ and define $x|z \sim p(x; \theta = f(Wz+b))$, where p is some density with parameters θ. $f(\cdot)$ induces coupling between the latent variables, which we can see by taking a Taylor expansion of some element of $f(z)$, $f_i(z)$, at z_0:

$$f_i(z) = f_i(z_0) + \nabla f_i(z_0)^T(z - z_0) + \frac{1}{2}(z - z_0)^T\nabla^2 f_i(z_0)(z - z_0) + \cdots . \tag{1.13}$$

The nonlinear coupling of latent variables endows the model with a great amount of flexibility, enabling the latent variables to represent distinct features of the data. Additionally, since the composition of two nonlinear functions does not reduce in the same way that linear functions do, we can add more latent variables to the model and 'stack' layers such that, e.g., $p(x) = p(x|\theta = f(W^{(1)}z^{(1)} + b^{(1)}))$ and $p(z^{(1)}) = p(z^{(1)}|\theta = f(W^{(0)}z^{(0)} + b^{(0)})$. The price we must pay for this flexibility is that $p(z|x)$ is no longer tractable.

Instead, variational inference (VI) [Jordan et al., 1999] introduces a variational distribution $q(\cdot)$ to approximate the posterior distribution and maximizes a lower bound on the evidence (ELBO):

$$\mathcal{L}(x) = \mathbb{E}_{q(z)}[\log p(x, z) - \log q(z)]. \tag{1.14}$$

It turns out that the gap between the ELBO and the evidence is the KL-divergence between the variational distribution and the true posterior:

$$\begin{aligned}
\log p(x) &= \int \log p(x) q(z) dz \\
&= \int [\log p(x, z) - \log p(z|x)] q(z) dz \\
&= \int [\log p(x, z) - \log q(z) + \log q(z) - \log p(z|x)] q(z) dz \\
&= \mathcal{L}(x) + KL(q(z)||p(z|x)).
\end{aligned} \tag{1.15}$$

If the variational distribution takes a particular form, such as a mean-field distribution $q(z) = \prod_i q_i(z)$ and $q_i(\cdot)$ is in the exponential family, coordinate ascent methods to optimize each $q_i(\cdot)$ individually can be performed. However, this can place a large constraint on the family of variational distributions available, potentially increasing the gap between the ELBO and the evidence.

We can improve the variational distribution's flexibility by amortizing $q(z)$ and turning it into a function of x, $q(z|x)$, and optimizing the ELBO through stochastic gradient descent [Bottou, 2010]. However, the training gradient from these estimates can be relatively noisy, often requiring design of control variates [Mnih and Gregor, 2014; Mnih and Rezende, 2016] to reduce the noise.

We can substantially reduce the noise in the ELBO gradient making use of the 'reparameterization trick' in variational autoencoders [Kingma and Welling, 2014; D.J. Rezende, 2014]. When the stochastic latent variable comes from a distribution that can be reparameterized as a function of parameterless 'base' distribution and parameters of the original latent distribution (e.g., $z \sim \mathcal{N}(\mu, \sigma^2) = \sigma * \epsilon + \mu$, $\epsilon \sim \mathcal{N}(0, I)$), we can express the ELBO gradient in terms of the latent parameters, which often greatly reduces its variance. This, however, constrains our latent variables to come from a class of parametric models.

1.5 Undirected models

An alternative to variational inference in directed models is the undirected energy-based model, in which inference is exact and easy, but sampling (which is exact and easy in directed models) is difficult.

The density of an energy-based model takes the form

$$p(x, z) = \frac{1}{Z} \exp(-E(x, z)), \tag{1.16}$$

where the energy function $E(x, z)$ entirely determines the model.

Note that we do not have to specify a prior distribution on the latents, in contrast to Factor Analysis and the directed model we will use for the variational autoencoder. It is tempting to view the prior distribution as trivial; however, we will see in Chapter 3 that this is the fundamental 'issue' with the autoencoder.

In Chapter 2, we will train the Restricted Boltzmann Machine whose energy function takes the form

$$E_{\text{RBM}}(x, z) = -a^T x - b^T z - x^T W z. \tag{1.17}$$

Evaluating the conditional distributions $p(x|z)$ and $p(z|x)$ is easy in this model. To perform inference, a simple calculation gives

$$p(z|x) = \prod_{i=1}^{P} p(z_i|x) = \prod_{i=1}^{P} \sigma(a_i + W_i^T z), \tag{1.18}$$

where $\sigma(\cdot)$ is the sigmoid function $\sigma(x) \triangleq \frac{1}{1+\exp(-x)}$.

The downside to this model is that it is very difficult to produce samples from the model. In contrast to directed models, in which one simply samples from the prior distribution over latent variables and performs a single feedfoward pass to sample x, in the RBM, one must perform (blockwise) Gibbs sampling by starting at some initial configuration $x^{(0)}$ and alternatively sampling $z^{(0)} \sim p(z|x^{(0)})$, $x^{(1)} \sim p(x|z^{(0)})$, $z^{(1)} \sim p(z|x^{(1)})$, ..., $x^{(N)} \sim p(x|z^{(N-1)})$, where N can be of the order of 10^5 to achieve proper burn-in [Salakhutdinov, 2010]. Since sampling from $p(x)$ is difficult, training has traditionally been done with approximate maximum likelihood methods, e.g, contrastive divergence [Hinton, 2002], which truncate the chain before it has reached convergence to the invariant distribution.

Chapter 2

Partition functions from Rao-Blackwellized tempered sampling

David Carlson*, Patrick Stinson*, Ari Pakman*, Liam Paninski, ICML 2015

(*equal contribution)

Partition functions of probability distributions are important quantities for model evaluation and comparisons. We present a new method to compute partition functions of complex and multimodal distributions. Such distributions are often sampled using simulated tempering, which augments the target space with an auxiliary inverse temperature variable. Our method exploits the multinomial probability law of the inverse temperatures, and provides estimates of the partition function in terms of a simple quotient of Rao-Blackwellized marginal inverse temperature probability estimates, which are updated while sampling. We show that the method has interesting connections with several alternative popular methods, and offers some significant advantages. In particular, we empirically find that the new method provides more accurate estimates than Annealed Importance Sampling when calculating partition functions of large Restricted Boltzmann

Machines (RBM); moreover, the method is sufficiently accurate to track training and validation log-likelihoods during learning of RBMs, at minimal computational cost.

2.1 Introduction

The computation of partition functions (or equivalently, normalizing constants) and marginal likelihoods is an important problem in machine learning, statistics and statistical physics, and is necessary in tasks such as evaluating the test likelihood of complex generative models, calculating Bayes factors, or computing differences in free energies. There exists a vast literature exploring methods to perform such computations, and the popularity and usefulness of different methods change across different communities and domain applications. Classic and recent reviews include [Gelman and Meng, 1998; Vyshemirsky and Girolami, 2008; Marin and Robert, 2009; Friel and Wyse, 2012].

In this paper we are interested in the particularly challenging case of highly multimodal distributions, such as those common in machine learning applications [Salakhutdinov and Murray, 2008]. Our major novel insight is that simulated tempering, a popular approach for sampling from such distributions, also provides an essentially cost-free way to estimate the partition function. Simulated tempering allows sampling of multimodal distributions by augmenting the target space with a random inverse temperature variable and introducing a series of tempered distributions. The idea is that the fast MCMC mixing at low inverse temperatures allows the Markov chain to land in different modes of the low-temperature distribution of interest [Marinari and Parisi, 1992; Geyer and Thompson, 1995].

As it turns out, (ratios of) partition functions have a simple expression in terms of ratios of the parameters of the multinomial probability law of the inverse temperatures. These parameters can be estimated efficiently by averaging the conditional probabilities of the inverse temperatures

along the Markov chain. This simple method matches state-of-the-art performance with minimal computational and storage overhead. Since our estimator is based on Rao-Blackwellized marginal probability estimates of the inverse temperature variable, we denote it Rao-Blackwellized Tempered Sampling (RTS).

In Section 2.2 we review the simulated tempering technique and introduce the new RTS estimation method. In Section 2.3, we compare RTS to Annealed Importance Sampling (AIS) and Reverse Annealed Importance Sampling (RAISE) [Neal, 2001; Burda et al., 2015], two popular methods in the machine learning community. We also show that RTS has a close relationship with Multistate Bennett Acceptance Ratio (MBAR) [Shirts and Chodera, 2008; Liu et al., 2015] and Thermodynamic Integration (TI) [Gelman and Meng, 1998], two methods popular in the chemical physics and statistics communities, respectively. In Section 2.4, we illustrate our method in a simple Gaussian example and in a Restricted Boltzmann Machine (RBM), where it is shown that RTS clearly dominates over the AIS/RAISE approach. We also show that RTS is sufficiently accurate to track training and validation log-likelihoods of RBMs during learning, at minimal computational cost. We conclude in Section 2.5.

2.2 Partition functions from tempered samples

In this section, we start by reviewing the tempered sampling approach. We then introduce our procedure to estimate partition functions by tempered sampling. We note here that our approach is useful not only as a stand-alone method for estimating partition functions, but is also essentially free in any application using tempered sampling.

2.2.1 Simulated tempering

Consider an unnormalized, possibly multimodal distribution proportional to $f(x)$, whose partition function we want to compute. Our method is based on simulated tempering, a well known approach to sampling multimodal distributions [Marinari and Parisi, 1992; Geyer and Thompson, 1995]. Simulated tempering begins with a normalized and easy-to-sample distribution $p_1(x)$ and augments the target distribution with a set of discrete inverse temperatures $\{0 = \beta_1 < \beta_2 < ... < \beta_K = 1\}$ to create a series of intermediate distributions between $f(x)$ and $p_1(x)$, given by

$$p(x|\beta_k) = \frac{f_k(x)}{Z_k}\,, \tag{2.1}$$

where

$$f_k(x) = f(x)^{\beta_k} p_1(x)^{1-\beta_k}\,, \tag{2.2}$$

and

$$Z_k = \int f_k(x)dx\,. \tag{2.3}$$

Z_K is the normalizing constant that we want to compute. Note that we assume $Z_1 = 1$ and $p(x|\beta_1) = p_1(x)$. However, our method does not depend on this assumption. When performing model comparison through likelihood ratios or Bayes factors, both distributions $f(x)$ and $p_1(x)$ can be unnormalized, and one is interested in the ratio of their partition functions. For the sake of simplicity, we consider here only the interpolating family given in (2.2); other possibilities can be used for particular distributions, such as moment averaging [Grosse et al., 2013] or tempering by subsampling [van de Meent et al., 2014].

When $\beta \in \{\beta_k\}_{k=1}^K$ is treated as a random variable, one can introduce a prior distribution

$r(\beta_k) = r_k$, and define the joint distribution

$$p(x, \beta_k) = p(x|\beta_k) r_k \tag{2.4}$$

$$= \frac{f_k(x) r_k}{Z_k}, \tag{2.5}$$

where Z_k is unknown. Instead, suppose we know approximate values $\hat{Z}_k$. Then we can define

$$q(x, \beta_k) \propto \frac{f_k(x) r_k}{\hat{Z}_k}, \tag{2.6}$$

which approximates $p(x, \beta_k)$. We note that the distribution q depends explicitly on the parameters $\hat{Z}_k$. A Gibbs sampler is run on this distribution by alternating between samples from $x|\beta$ and $\beta|x$. In particular, the latter is given by

$$q(\beta_k|x) = \frac{f_k(x) r_k / \hat{Z}_k}{\sum_{k'=1}^{K} f_{k'}(x) r_{k'} / \hat{Z}_{k'}}. \tag{2.7}$$

Sampling as such enables the chain to traverse the inverse temperature ladder stochastically, escaping local modes under low β and collecting samples from the target distribution $f(x)$ when $\beta = 1$ [Marinari and Parisi, 1992].

2.2.2 Estimating partition functions

Letting $\hat{Z}_1 \equiv Z_1 = 1$, we first note that by integrating out x in (2.6) and normalizing, the marginal distribution over the β_k's is

$$q(\beta_k) = \frac{r_k Z_k / \hat{Z}_k}{\sum_{k'=1}^{K} r_{k'} Z_{k'} / \hat{Z}_{k'}}. \tag{2.8}$$

Note that if $\hat{Z}_k$ is not close to Z_k for all k, the marginal probability $q(\beta_k)$ will differ from the prior r_k, possibly by orders of magnitude for some k's, and the β_k's will not be efficiently sampled. One approach to compute approximate $\hat{Z}_k$ values is the Wang-Landau algorithm [Wang and Landau, 2001; Atchade and Liu, 2010]. We use an iterative strategy, discussed in Section 2.2.4.

Given samples $\{x^{(i)}, \beta_{k^{(i)}}\}$ generated from $q(x, \beta_k)$, the marginal probabilities above can simply be estimated by the normalized counts for each bin β_k, $\frac{1}{N}\sum_{i=1}^{N}\delta_{k,k^{(i)}}$. But a lower variance estimator can be obtained by the Rao-Blackwellized [Robert and Casella, 2013] form

$$\hat{c}_k = \frac{1}{N}\sum_{i=1}^{N} q(\beta_k|x^{(i)})\,. \tag{2.9}$$

Note that our estimates in (2.9) are unbiased estimators of (2.8), since

$$q(\beta_k) = \int q(\beta_k|x)q(x)dx\,. \tag{2.10}$$

Our main idea is that the exact partition function can be expressed by ratios of the marginal distribution in (2.8),

$$Z_k = \hat{Z}_k \frac{r_1}{r_k}\frac{q(\beta_k)}{q(\beta_1)} \qquad k = 2, \ldots, K\,. \tag{2.11}$$

Plugging our estimates $\hat{c}_k$ of $q(\beta_k)$ into (2.11) immediately gives us the consistent estimator

$$\hat{Z}_k^{\mathrm{RTS}} = \hat{Z}_k \frac{r_1}{r_k}\frac{\hat{c}_k}{c_1} \qquad k = 2, \ldots, K\,. \tag{2.12}$$

The resulting procedure is outlined in Algorithm 2.

2.2.3 Rao-Blackwellized likelihood interpretation

We can alternatively derive (2.12) by optimizing a Rao-Blackwellized form of the marginal likelihood. From (2.8), the log-likelihood of the $\{\beta_{k^{(i)}}\}$ samples is

$$\begin{aligned}\log q(\{\beta_{k^{(i)}}\}_{i=1}^{N}) = &\sum_{i=1}^{N}\log(Z_{k^{(i)}}) \\ &- N\log\left(\sum_{k=1}^{K} r_k Z_k/\hat{Z}_k\right) + const.\end{aligned} \tag{2.13}$$

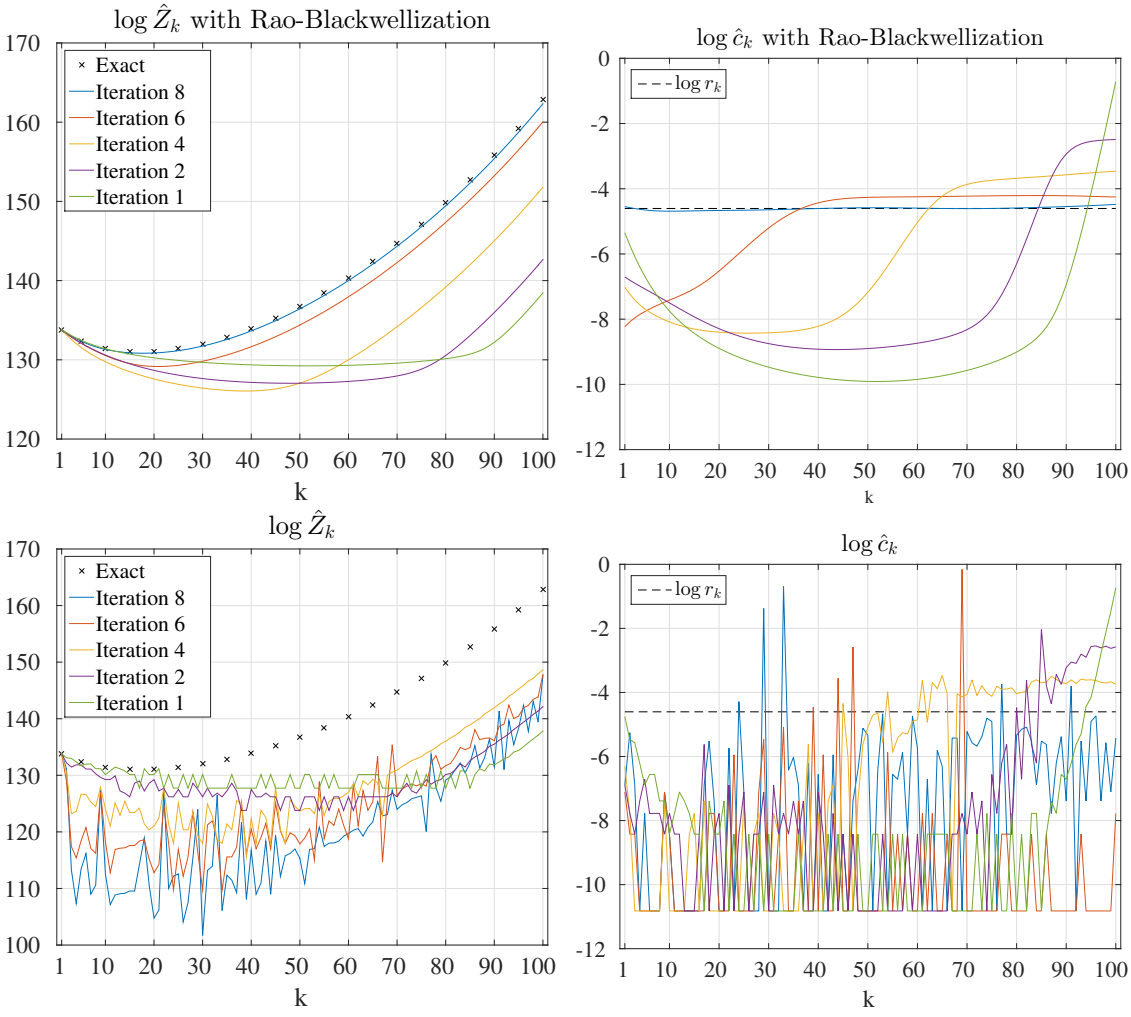


Figure 2.1: Comparison of $\log \hat{Z}_k$ and $\log \hat{c}_k$ estimates, in some of the first eight iterations of the initialization procedure described in Section 2.2.4, with and without Rao-Blackwellization, with $K = 100$. The initial values were $\hat{Z}_k = 1$ for all k, and the prior was uniform, $r_k = 1/K$. The model is a RBM with 784 visible and 10 hidden units, trained on the MNIST dataset. Each iteration consists of 50 Gibbs sweeps, on each of 100 parallel chains. Since in the non-Rao-Blackwellized case, the updates are unstable and sometimes infinite, for demonstration purposes only, we define $\hat{c}_k \propto 0.1 + \sum_{i=1}^{N} \delta_{k,k^{(i)}}$ and normalize. Note that in the Rao-Blackwellized case, the values of $\hat{c}_k$ in the final iteration are very close to those of r_k, signaling that the $\hat{Z}_k$'s are good enough for a last, long MCMC run to obtain the final $\hat{Z}_k$ estimates.

Algorithm 2 Rao-Blackwellized Tempered Sampling

Input: $\{\beta_k, r_k\}_{k=1,...,K}$, N

Initialize $\log \hat{Z}_k,\ k = 2, ..., K$

Initialize $\beta \in \{\beta_1, ..., \beta_K\}$

Initialize $\hat{c}_k = 0,\ k = 1, ..., K$

for $i = 1$ **to** N **do**

Transition in x leaving $q(x|\beta)$ invariant.

Sample $\beta|x \sim (\beta|x)$

Update $\hat{c}_k \leftarrow \hat{c}_k + \frac{1}{N} q(\beta_k|x)$

end for

Update $\hat{Z}_k^{\mathrm{RTS}} \leftarrow \hat{Z}_k \frac{r_1 \hat{c}_k}{r_k \hat{c}_1},\ k = 2, ..., K$

Because $\beta_{k^{(i)}}$ was sampled from $q(\beta|x^{(i)})$, we can reduce variance by Rao-Blackwellizing the first sum in (2.13), resulting in

$$\begin{aligned} L_{RB}[\mathbf{Z}] &= \sum_{i=1}^{N} \sum_{k=2}^{K} \log(Z_k) q(\beta_k|x^{(i)}) \\ &\quad - N \log\left(\sum_{k=1}^{K} r_k Z_k / \hat{Z}_k \right) + const, \\ &= N \sum_{k=2}^{K} \log(Z_k) \hat{c}_k \\ &\quad - N \log\left(\sum_{k=1}^{K} r_k Z_k / \hat{Z}_k \right) + const\,. \end{aligned} \tag{2.14}$$

The normalizing constants are estimated by maximizing (2.14) subject to a fixed Z_1, which is known. Setting the derivatives of (2.14) w.r.t. Z_k's to zero gives a system of linear equations

$$\sum_{k'=2}^{K} \frac{r_{k'}}{\hat{Z}_{k'}} \left(\frac{\delta_{k',k}}{\hat{c}_k} - 1 \right) Z_{k'} = r_1 \qquad k = 2, \ldots, K$$

whose solution is (2.12).

2.2.4 Initial iterations

As mentioned above, the chain with initial $\hat{Z}_k$'s may mix slowly and provide a poor estimator (i.e. small $q(\beta_k)$'s are rarely sampled). Therefore, when the $\hat{Z}_k$'s are far from the Z_k's (or equivalently, the r_k's are far from the $\hat{c}_k$'s), the $\hat{Z}_k$'s estimates should be updated.

Our estimator in (2.12) does not directly handle the case where $\hat{Z}_k$ is sequentially updated. We note that the likelihood approach of (2.14) is straightforwardly adapted to this case and is straightforwardly numerically optimized (see Section 2.4.4 for details). A simpler, less computationally intensive, and equally effective strategy is as follows: start with $\hat{Z}_k = 1$ for all k (or a better estimate, if known), and iterate between estimating $\hat{c}_k$ with few MCMC samples and updating $\hat{Z}_k$ with the estimated $\hat{Z}_k^{\mathrm{RTS}}$ using (2.12). In our experiments using many parallel Markov chains, this procedure worked best when the updated Markov chains started from the previous last x's, and fresh, uniformly random sampled β_k's.

Once the $\hat{Z}_k$'s estimates are close enough to the Z_k's to facilitate mixing, a long MCMC chain can be run to provide samples for the estimator. Because $\hat{c}_k$ estimates $q(\beta_k)$, and $q(\beta_k) \simeq r_k$ when $\hat{Z}_k \simeq Z_k$, a simple stopping criterion for the initial iterations is to check the similarity between $\hat{c}_k$ and r_k. For example, if we use a uniform prior $r_k = 1/K$, a practical rule is to iterate the few-samples chains until $\max_k |r_k - \hat{c}_k| < 0.1/K$.

Figure 2.1 shows the values taken by $\hat{Z}_k$ and $\hat{c}_k$ in these initial iterations in a simple example. The figure also illustrates the importance of using the Rao-Blackwellized form (2.9) for $\hat{c}_k$, which dramatically reduces the noise in the estimator $\frac{1}{N}\sum_{i=1}^{N} \delta_{k,k^{(i)}}$ for $q(\beta_k)$.

2.2.5 Bias and variance

Using (2.11)-(2.12) and $\log(1+x) \simeq x - x^2/2$, gives

$$\log \hat{Z}_k^{\mathrm{RTS}} \approx \log Z_k + \frac{\Delta c_k}{q_k} - \frac{\Delta c_1}{q_1} - \frac{(\Delta c_k)^2}{2q_k^2} + \frac{(\Delta c_1)^2}{2q_1^2} \tag{2.15}$$

where $q_k = q(\beta_k)$ and $\Delta c_k = \hat{c}_k - q_k$. Taking expectations gives

$$\mathbb{E}\left[\log \hat{Z}_k^{\mathrm{RTS}}\right] - \log Z_k \approx \frac{1}{2}\left[\frac{\sigma_1^2}{\hat{c}_1^2} - \frac{\sigma_k^2}{\hat{c}_k^2}\right], \tag{2.16}$$

and

$$\mathrm{Var}[\log \hat{Z}_k^{\mathrm{RTS}}] \approx \frac{\sigma_1^2}{\hat{c}_1^2} + \frac{\sigma_k^2}{\hat{c}_k^2} - \frac{2\sigma_{1k}}{\hat{c}_k \hat{c}_1} \tag{2.17}$$

where $\sigma_1^2 = \mathrm{Var}[\hat{c}_1]$, $\sigma_k^2 = \mathrm{Var}[\hat{c}_k]$, and $\sigma_{1k} = \mathrm{Cov}[\hat{c}_1, \hat{c}_k]$.

This shows that the bias of $\log \hat{Z}_k$ has no definite sign. This is in contrast to many popular methods, such as AIS, which underestimates $\log Z_k$ [Neal, 2001], and RAISE, which overestimates $\log Z_k$ [Burda et al., 2015].

From the Central Limit Theorem, the asymptotic variance of $\hat{c}_k$ is

$$Var(\hat{c}_k) = \frac{Var_q(q(\beta_k|x))a_k}{N}, \tag{2.18}$$

where the factor

$$a_k = 1 + 2\sum_{i=1}^{\infty} \mathrm{corr}\left[q(\beta_k|x^{(0)}), q(\beta_k|x^{(i)})\right] \tag{2.19}$$

takes into account the autocorrelation of the Markov chain. But estimates of this sum from the MCMC samples are generally too noisy to be useful. A more practical approach is to estimate $Var[\hat{c}_k]$ from $\hat{c}_k$ estimates of many parallel MCMC chains.

2.3 Related work

In this section, we briefly review some popular estimators and explore their relationship to the proposed RTS estimator (2.12).

2.3.1 Wang-Landau

A well-known approach to obtain approximate values of the Z_k's is the Wang-Landau algorithm [Wang and Landau, 2001; Atchade and Liu, 2010]. The setting is similar to ours, but the algorithm constantly modifies the $\hat{Z}_k$'s along the Markov chain as different β_k's are sampled. The factors that change the $\hat{Z}_k$'s asymptotically converge to 1. The resulting $\hat{Z}_k$ estimates are usually good enough to allow mixing in the (x, β) space [Salakhutdinov, 2010], but are too noisy for purposes such as likelihood estimation [Tan, 2016].

2.3.2 AIS/RAISE

Annealed Importance Sampling (AIS) [Neal, 2001] is perhaps the most popular method in the machine learning literature to estimate $\log Z_K$. Here, one starts from a sample x_1 from $p_1(x)$, and samples a point x_2, using a transition function $K_2(x_2|x_1)$ that leaves $f_2(x)$ invariant. The process is repeated until one has sampled x_K using a transition function that leaves $f(x)$ invariant. The vector $(x_1, x_2, ..., x_K)$ is interpreted as a sample from an importance distribution on an extended space, while the original distribution $p(x_K)$ can be similarly augmented into an extended space. The resulting importance weight can be computed in terms of quotients of the f_k's, and provides an unbiased estimator for Z_K/Z_1, whose variance decreases linearly with K. Note that the inverse temperatures in this approach are not random variables.

The variance of the AIS estimator can be reduced by averaging over several runs, but the resulting value of $\log(\hat{Z}_K)$ has a negative bias due to Jensen's inequality. This in turn results in a

positive bias when estimating data log-likelihoods.

Recently, a related method, called Reverse Annealed Importance Sampling (RAISE) was proposed to estimate the data log-likelihood in models with latent variables, giving negatively biased estimates [Burda et al., 2015], [Z. et al., 2015]. The method performs a similar sampling as AIS, but starts from a sample of the latent variables at $\beta_K = 1$ and proceeds then to lower inverse temperatures. In certain cases, such as in the RBM examples we consider in Section 2.4.2, one can obtain from these estimates of the data log-likelihood an estimate of the partition function, which will have a positive bias. The combination of the expectations of these AIS and RAISE estimators thus 'sandwiches' the exact value [Burda et al., 2015], [Z. et al., 2015].

2.3.3 BAR/MBAR

Bennett's acceptance ratio (BAR) [Bennett, 1976], also called bridge sampling [X.-L.Meng and Wong, 1996], is based on the identity

$$\frac{Z_k}{Z_1} = \frac{\mathbb{E}_{p(x|\beta_1)}[\alpha(x) f_k(x)]}{\mathbb{E}_{p(x|\beta_k)}[\alpha(x) f_1(x)]}\,, \tag{2.20}$$

where $\alpha(x)$ is an arbitrary function such that $\int f_1(x) f_k(x) \alpha(x) dx < \infty$, which can be chosen to minimize the asymptotic variance. BAR has been generalized to estimate partition functions when sampling among multiple distributions, a method termed the multistate BAR (MBAR) [Shirts and Chodera, 2008].

Assuming that there are n_k i.i.d. samples for each inverse temperature β_k (N samples $\{x_i\}_{i=1,\ldots,N}$ in total), and $\Delta_x = \log f(x) - \log p_1(x)$, the MBAR partition function estimates can

be obtained by maximizing the log-likelihood function [Tan et al., 2012]:

$$L[\mathbf{Z}] = \frac{1}{N}\sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\frac{n_k}{N}\exp(-\log Z_k + \beta_k\Delta_{x_i})\right) + \sum_{r=1}^{K}\frac{n_r}{N}\log Z_r \tag{2.21}$$

This method was recently rediscovered and shown to compare favorably against AIS/RAISE in [Liu et al., 2015]. MBAR has many different names in different literatures, e.g. unbinned weighted histogram analysis method (UWHAM) [Tan et al., 2012] and reverse logistic regression [Geyer, 1994].

Unlike RTS, MBAR does explicitly use $q(\beta)$ when estimating the partition function. As a price associated with this increased generality, MBAR requires the storage of all collected samples, and the estimator is calculated by finding the maximum of (2.21). This likelihood function does not have an analytic solution, and Newton-Raphson was proposed to iteratively solve this problem, which requires $\mathcal{O}(NK^2 + K^3)$ per iteration. While RTS is less general than MBAR, RTS has an analytic solution and only requires the storage of the $\hat{c}_k$ statistics. We note that this objective function is very similar to the one discussed in Section 2.4.4 for pooling across samples collected using different $\hat{Z}_k$'s.

Recent work has proposed a stochastic learning algorithm based on MBAR/UWHAM [Tan et al., 2016]. This algorithm gives updates based on the sufficient statistics $\hat{c}_k$ with

$$\log\hat{Z}_k^{(t+1)} = \log\hat{Z}_k^{(t)} + \gamma_t\left(\frac{\hat{c}_k}{r_k} - \frac{\hat{c}_1}{r_1}\right). \tag{2.22}$$

γ_t is a step size that is recommended to be set to $\gamma_t = t^{-1}$. We note that our estimator from (2.12) in log space may be written in a similar form, as $\left(\log\left(\frac{\hat{c}_k}{r_k}\right) - \log\left(\frac{\hat{c}_1}{r_1}\right)\right)$, which is very related in form to (2.22). We empirically found that when the partition function estimates are far away from the truth, our update (2.12) dominates over (2.22). Because a first order approximation to our

estimator in (2.15) is the same as the term in (2.22), the updates will essentially only differ by the selection of the step size γ_t when $\hat{c}_k \simeq r_k$.

2.3.4 Thermodynamic integration

Thermodynamic Integration [Gelman and Meng, 1998] is derived from basic calculus identities. Let us first assume that β is a continuous variable in $[0, 1]$. We again define $\Delta_x = \log f(x) - \log p_1(x)$, and $f_\beta(x) = f(x)^\beta p_1(x)^{1-\beta}$. We note that

$$\begin{aligned} \frac{d}{d\beta} \log Z(\beta) &= \int \frac{1}{Z(\beta)} \frac{d}{d\beta} f_\beta(x) dx \\ &= \mathbb{E}_{x|\beta}[\Delta_x], \end{aligned} \tag{2.23}$$

From calculus, we have

$$\log \left(\frac{Z_K}{Z_1} \right) = \int_0^1 \mathbb{E}_{x|\beta}[\Delta_x] d\beta = \mathbb{E}_{p(x|\beta)p(\beta)} \left[\frac{\Delta_x}{p(\beta)} \right]$$

This equation holds for any $p(\beta)$ that is positive over the range $[0, 1]$, and provides an unbiased estimator for $\log Z_k$ if unbiased samples from $p(x|\beta)$ are available. This is in contrast to AIS, which is unbiased on Z_k, and biased on $\log Z_k$. Given samples $\{x^{(i)}, \beta^{(i)}\}_{i=1,\ldots,N}$, the estimator for $\log Z_K$ is

$$\widehat{\log Z_K} = \log Z_1 + \frac{1}{N} \sum_{i=1}^{N} \frac{\Delta_{x^(i)}}{p(\beta^{(i)})}$$

There are two distinct approaches for generating samples and performing this calculation in TI. First, β can be sampled from a prior $p(\beta)$, and samples are generated from $f_\beta(x)$ to estimate the gradient at the current point in β space. A second approach is to use samples generated from simulated tempering, which can facilitate mixing. However, the effective marginal distribution $q(\beta)$ must be estimated in this case.

When β consists of a discrete set of inverse temperatures, the integral can be approximated by the trapezoidal or Simpson's rule. Recently, higher order moments were used to improve this

integration, which can help in some cases [Friel et al., 2014]. As noted by [Calderhead and Girolami, 2009], this discretization error can be expressed as a sum of KL-divergences between neighboring intermediate distributions. If the KL-divergences are known, an optimal discretization strategy can be used. However, this is unknown in general.

While the point of this paper is not to improve the TI approach, we note that the Rao-Blackwellization technique we propose also applies to TI when using tempered samples. This gives that the Monte Carlo approximation of the gradient (2.23) is

$$\left.\frac{d}{d\beta}\log Z(\beta)\right|_{\beta=\beta_k} \simeq \sum_{i=1}^{N} \frac{q(\beta_k|x_i)\Delta_{x_i}}{\sum_{j=1}^{N} q(\beta_k|x_j)}. \tag{2.24}$$

This reduces the noise on the gradient estimates, and improves performance when the number of bins is relatively high compared to the number of collected samples. We refer to this technique as TI-Rao-Blackwell (TI-RB).

TI-RB is further interesting in the context of RTS, because of a surprising relationship: in the continuous β limit, RTS and TI-RB are *equivalent* estimators. However, when using discrete inverse temperatures, RTS does not suffer from the discretization error that TI and TI-RB do.

2.4 Examples

In this section, we study the ability of RTS to estimate partition functions in a Gaussian mixture model and in Restricted Boltzmann Machines and compare to estimates from popular existing methods. We also study the dependence of several methods on the number K of inverse temperatures, and show that RTS can provide estimates of train- and validation-set likelihoods during RBM training at minimal cost.

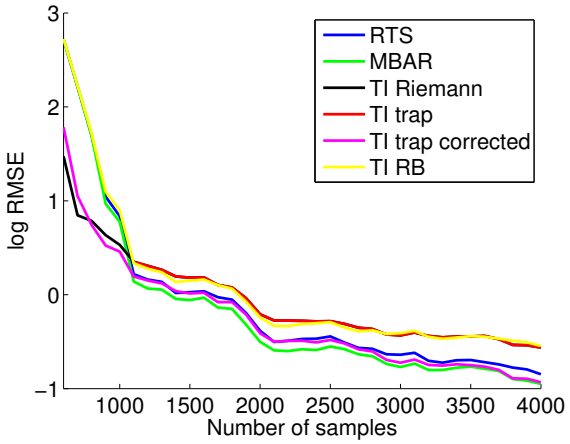


Figure 2.2: Comparison of $\log Z$ estimation performance on a toy Gaussian Mixture Model using an RMSE from 10 repeats. TI Riemann approximates the discrete integral as a right Riemann sum, TI trap uses the trapezoidal method, TI trap corrected uses a variance correction technique, TI RB uses the Rao-Blackwellized version of TI.

2.4.1 Gaussian mixture example and comparisons

Figure 2.2 compares the performance of RTS to several methods, including MBAR and TI and its variants, in a mixture of two 10-dimensional Gaussians (see Section A.2 for specific details). The sampling was performed using a novel adaptive Hamiltonian Monte Carlo method for tempered distributions of continuous variables, introduced in Section A.2. In this case the exact partition function can be numerically estimated to high precision. Note that the estimators essentially give identical performance; however, our method is the simplest to implement and use for tempered samples, with minimal memory and computation requirements.

2.4.2 Partition functions of RBMs

The Restricted Boltzmann Machine (RBM) is a bipartite Markov Random Field model popular in the machine learning community [Smolensky, 1986]. For the binary case, this is a generative model over visible observations $v \in \{0,1\}^M$ and latent features $h \in \{0,1\}^J$ defined by $\log f(v,h) =$

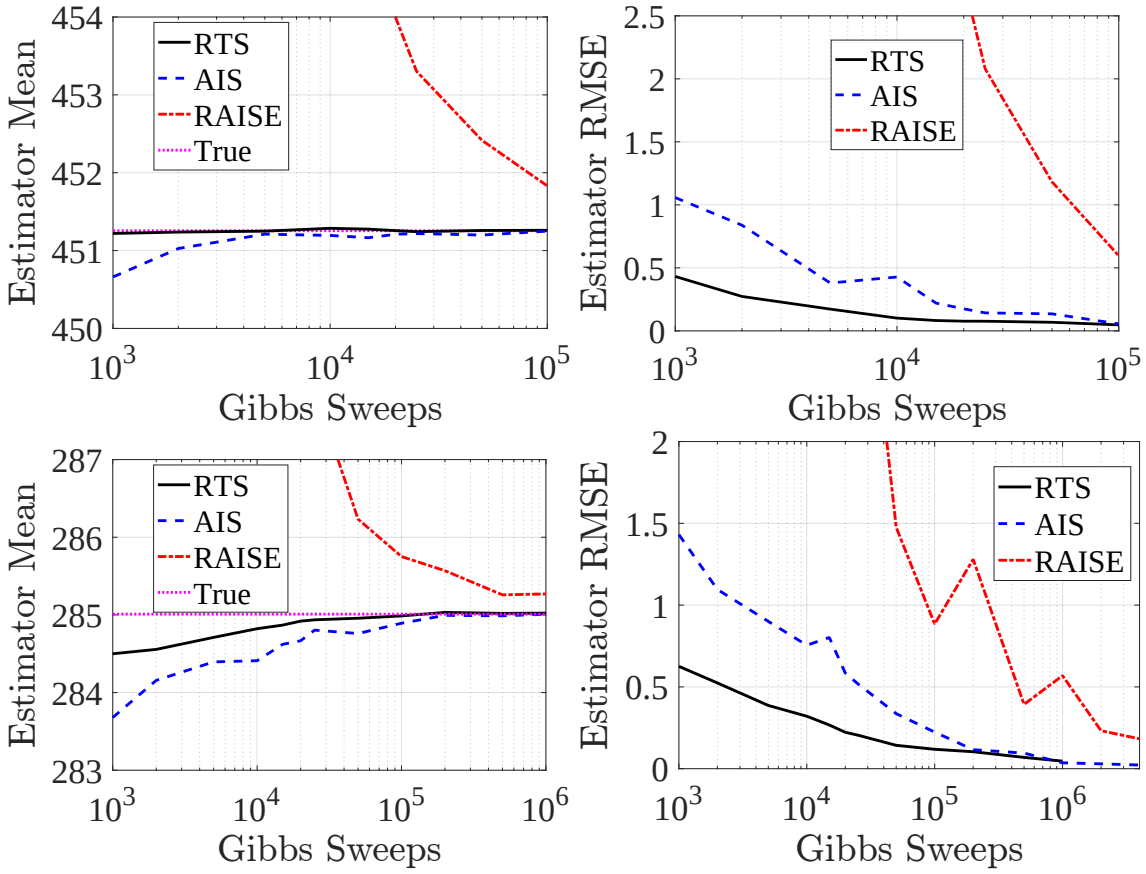


Figure 2.3: Mean and root mean squared error (RMSE) of competing estimators of $\log Z_K$ evaluated on RBMs with 784 visible units trained on the MNIST dataset. The numbers of hidden units were 500 (Top) and 100 (Bottom). In both cases, the bias from RTS decreases quicker than that of AIS and RAISE, and the RMSE of AIS does not approach that of RTS at 1000 Gibbs sweeps until over an order of magnitude later. Each method is run on 100 parallel Gibbs chains, but the Gibbs sweeps in the horizontal axis corresponds to each individual chain.

$v^T c + v^T W h + h^T b$, for parameters $c \in \mathbb{R}^M$, $b \in \mathbb{R}^J$, and $W \in \mathbb{R}^{M \times J}$. A fundamental performance measure of this model is the log-likelihood of a test set, which requires the estimation of the log partition function. Both AIS [Salakhutdinov and Murray, 2008] and RAISE [Burda et al., 2015] were proposed to address this issue. We will evaluate performance on the bias and the root mean squared error (RMSE) of the estimator. To estimate "truth," we estimate the true mean as the

average of estimates from AIS and RTS with 10^6 samples from 100 parallel chains. We note the variance of these estimates was very low (≈ 0.006).

Figure 2.3 shows a comparison of RTS versus AIS/RAISE on two RBMs trained on the binarized MNIST dataset (M=784, N=60000), with 500 and 100 hidden units. The former was taken from [Salakhutdinov and Murray, 2008],[1] while the latter was trained with the method of [Carlson et al., 2015].

In all the cases we used for p_1 a product of Bernoulli distributions over the v variables which matches the marginal statistics of the training dataset, following [Salakhutdinov and Murray, 2008]. We run each method (RTS, AIS, RAISE) with 100 parallel Gibbs chains. In RTS, the number of inverse temperatures was fixed at K=100, and we performed 10 initial iterations of 50 Gibbs sweeps each, following Section 2.2.4. In AIS/RAISE, the number of inverse temperatures K was set to match in each case the total number of Gibbs sweeps in RTS, so the comparisons in Figure 2.3 correspond to matched computational costs. We note that the performance of RAISE is similar to the plots shown in [Burda et al., 2015] for these parameters.

2.4.3 Number of temperatures

An advantage of the Rao-Blackwellization of temperature information is that there is no need to pick a precise number of inverse temperatures, as long as K is big enough to allow for good mixing of the Markov chain. As shown in Figure 2.4, RTS's performance is not greatly affected by adding more temperatures once there are enough temperatures to give good mixing.

Also note that as the number of temperatures increases RTS and the Rao-Blackwellized version of TI (TI-RB) become increasingly similar. We show explicitly in Section A.4 that they are equivalent in the infinite limit of the number of temperatures. Due to computational costs, running

[1] Code and parameters available from: `http://www.cs.toronto.edu/~rsalakhu/rbm_ais.html`

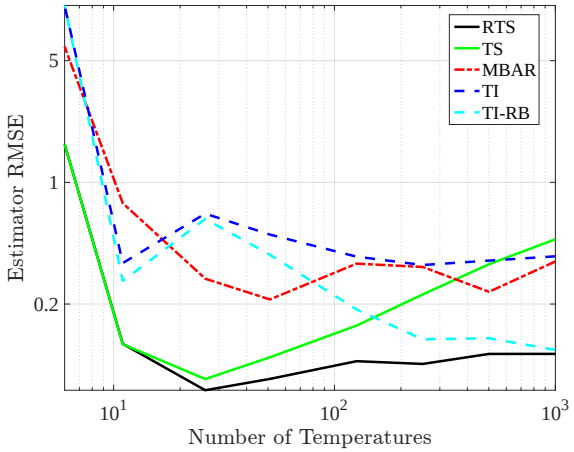


Figure 2.4: RMSE as a function of the number of inverse temperatures K for various estimators. The model is the same RBM with 500 hidden units studied in Figure 2.3. Each point was obtained by averaging over 200 estimates (20 for MBAR due to computational costs) made from 10,000 bootstrapped samples from a long MCMC run of 3 million samples.

MBAR on a large number of temperatures is computationally prohibitive. An issue when estimates are non-Rao-Blackwellized is that the estimates eventually become unstable as we do not have positive counts for each bin. This is addressed heuristically in the non-Rao-Blackwellized version of RTS (TS) by adding a constant of .1 to each bin. For TI, empty bins are imputed by linear interpolation.

2.4.4 Tracking partition functions while training

There are many approaches to training RBMs, including recent methods that do not require sampling [Sohl-Dickstein et al., 2010; Im et al., 2015; Gabrie et al., 2015]. However, most learning algorithms are based on Monte Carlo Integration with persistent Contrastive Divergence [Tieleman and Hinton, 2009]. This includes proposals based on tempered sampling [Salakhutdinov, 2009; Desjardins et al., 2010]. In these cases, the slow speed of change of the parameters and the relatively low number of samples required by RTS, allow us to track the value of a train- and validation-set

likelihoods during RBM training at minimal additional cost. This allows us to avoid overfitting by early stopping of the training. We note that there are previous more involved efforts to track RBM partition functions, which involve additional computational and implementation efforts [Desjardins et al., 2011].

This idea is illustrated in Figure 2.5, which shows estimates of the mean of training and validation log-likelihoods on the *dna* dataset[2], with 180 observed binary features, trained on a RBM with 500 hidden units.

We first pretrain the RBM with CD-1 to get initial values for the RBM parameters. We then run initial RTS iterations with $K = 100$, as in Section 2.2.4, in order to get starting $\log \hat{Z}_k$ estimates.

For the main training effort we used the RMSspectral gradient method, with stepsize of 1e-5 and parameter $\lambda = .99$ (see [Carlson et al., 2015] for details). We considered a tempered space with $K = 100$ and sampled 25 Gibbs sweeps on 2000 parallel chains between gradient updates. The latter is a large number compared to older learning approaches [Salakhutdinov and Murray, 2008], but is similar to that used both in [Carlson et al., 2015] and [Grosse and Salakhudinov, 2015] that provide state-of-the-art learning techniques.

With the samples collected after each 25 Gibbs sweeps, we can estimate the $\hat{c}_k$'s to compute the running partition function. To smooth the noise from such a small number of samples, we consider partial updates of $\hat{Z}_K$ given by

$$\hat{Z}_K^{(t+1)} = \hat{Z}_K^{(t)} \left(\frac{r_1}{r_K} \frac{\hat{c}_K^{(t)}}{\hat{c}_1^{(t)}} \right)^{\alpha} \tag{2.25}$$

with $\alpha = 0.2$, and t an index on the gradient update. Similar results were obtained with $.05 < \alpha < .5$. This smoothing is also justified by the slowly changing nature of the parameters. Figure 2.5 also

[2]Available from: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

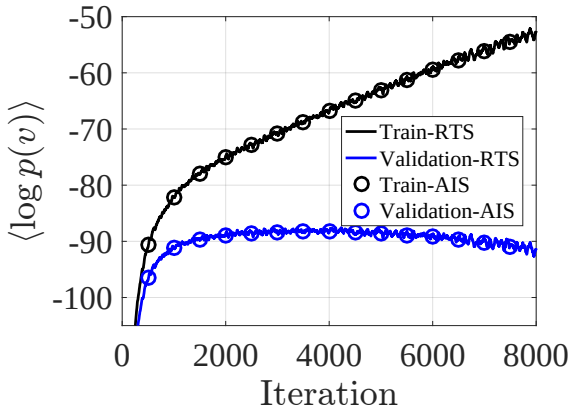


Figure 2.5: A demonstration of the ability to track with minimal cost the mean train and validation log-likelihood during the training of a RBM on the *dna* 180-dimensional binary dataset, with 500 latent features.

shows the corresponding value from AIS with 100 parallel samples and 10,000 inverse temperatures. Such AIS runs have been shown to give accurate estimates of the partition function for RBMs with even more hidden units [Salakhutdinov and Murray, 2008], but involve a major computational cost that our method avoids. Using the settings from [Salakhutdinov and Murray, 2008] adds a cost of 10^6 additional samples.

2.5 Discussion

In this paper, we have developed a new partition function estimation method that we called Rao-Blackwellized Tempered Sampling (RTS). Our experiments show RTS has equal or superior performance to existing methods popular in the machine learning and physical chemistry communities, while only requiring sufficient statistics collected during simulated tempering.

An important free parameter is the prior over inverse temperatures, r_k, and its optimal selection is a natural question. We explored several parametrized proposals for r_k, but in our experiments no distribution consistently performed significantly better than the uniform. We also explored a

continuous β formulation, but the resulting estimates were less accurate. Additionally, we tried subtracting off estimates of the bias, but this did not improve the results. Finally, we tried incorporating a variety of control variates, such as those in [Dellaportas and Kontoyiannis, 2012], but did not find them to reduce the variance of our estimates in the examples we considered. Other control variates methods, such as those in [Oates et al., 2015], could potentially be combined with RTS in continuous distributions.

Chapter 3

Decoupling aggregate priors in variational autoencoders

The choice of the generative model prior is an important part of designing variational autoencoders. The variational posterior averaged over the data distribution uniquely minimizes the evidence lower bound (ELBO) with respect to the prior; consequently, a popular prior choice is a direct estimate of the variational posterior by averaging a fixed number of encoding distributions. However, since the encoding model is regularized by the prior in the ELBO, such direct coupling of the prior and variational distribution leads to additional constraints on the encoding model, which can limit performance. We propose a generalization of the aggregate approximation prior by endowing it with generic 'delta' functions parameterized independently from the encoder, giving rise to a more flexible prior capable of decoupling from the encoder model, which we show improves the latent representation. We also show that when this approach is used in conjunction with a semi-implicit aggregate prior, it greatly improves performance and gives superior log-likelihoods compared to existing aggregate models. Finally, we draw a parallel between the decoupled semi-implicit model and kernel density estimation.

3.1 Introduction

Generative modeling, which aims to learn and produce samples from a dataset's underlying probability distribution, is a major goal of machine learning. Variational autoencoders (VAEs) [Kingma and Welling, 2014; D.J. Rezende, 2014] have become very popular over the past few years partly due to their combining inference and generative modeling into one framework, with the evidence lower bound (ELBO) on the marginal log-likelihood reflecting both the inference and generative models. Integral to its success is the reparameterization trick, which enables stochastic gradients of the ELBO to leverage the latent space density's parametric form, thereby reducing variance.

A common consequence of the latent distribution's parametric form is that it can be overly simplistic relative to the true posterior and cannot accurately approximate it, represented by the gap between the ELBO and the marginal log-likelihood. Consequently, efforts have been made to increase the variational distribution's expressiveness including using flow-based models [Rezende and Mohamed, 2015; Kingma et al., 2016; van den Berg et al., 2018; Chen et al., 2018], implicit variational models [Huszar, 2017; Mescheder et al., 2017; Tran et al., 2017; Yin and Zhou, 2018; Shi et al., 2018], adversarial models [Mescheder et al., 2017], and Bayesian nonparametric models [Tran et al., 2016; Nalisnick and Smyth, 2017]

However, due to being regularized by the generative model's prior, the variational distribution may not reach its full expressive capacity even under more sophisticated encoder models. Due to the nature of the KL-divergence penalty, the variational distribution will tend to avoid putting probability density in regions in which the prior's density is low [Ranganath et al., 2016], potentially limiting the variational model. Another drawback of overregularization of the variational distribution is that samples from the trained model may not be meaningful since the latent vari-

able is drawn from the prior's wider density [Makhzani et al., 2016]. When the decoder model is sophisticated enough, for example in autoregressive decoders (e.g., [van den Oord et al., 2016a; van den Oord et al., 2016b], the KL penalty may prevent the model from learning a useful encoding entirely [Alemi et al., 2017], and optimization heuristics must be used, including annealing the KL penalty at the start of training [Bowman et al., 2016; Sønderby et al., 2016; Serban et al., 2017] or effectively eliminating the KL penalty up to some quantity [Kingma et al., 2016]. Additionally, specific modeling constraints can be put on the decoder to require the latent space to be informative [Chen et al., 2017].

Designing the prior has received less attention than the variational distribution, perhaps due to its perceived relative simplicity, or that many of the methods used to increase the expressivity of the encoding model (e.g., flow-based and autoregressive models) can be used similarly for the prior. In this paper, we restrict our attention to models that use for the prior an approximation of the aggregate variational posterior that is either explicit [Tomczak and Welling, 2018] or (semi-) implicit [Molchanov et al., 2019]. An alternate expression of the ELBO by [Hoffman and Johnson, 2016] using a marginal KL penalty served as motivation for these models, as the penalty is minimized by the aggregate variational distribution. However, we argue that alone, these approximations can place unnecessary constraints on the encoder model and hinder overall performance. Thus, the aggregate approximation prior still has an effect on the encoding model, despite the technique's motivation to simply minimize one term in the ELBO. In fact, as we show empirically, a better prior can even increase the marginal KL if the reconstruction quality is sufficiently improved.

We propose a generic decoupling model to endow the prior with flexibility while still utilizing information about the encoder, which we show improves reconstruction and latent representation quality. Decoupling the semi-implicit prior in particular leads to superior test log-likelihoods over existing aggregate methods and is robust to changes in the granularity of the aggregate approxima-

tion. Finally, we draw a connection between the decoupled semi-implicit model and kernel density estimation.

3.2 Variational Autoencoders

Given some data, $\{x_n\}_{n=1}^N$, $x_n \sim p_{\text{true}}(x)$, latent variable models circumvent direct modeling of the observed data and instead assume a set of stochastic unobserved variables z interact according to $p(z)$ and influence the observed variables according to $p(x|z)$. However, the posterior probability $p(z|x)$ is often intractable to compute. Instead, variational inference [Jordan et al., 1999] introduces a variational distribution $q(z)$ that functions as a tractable approximation to the posterior distribution.

Without access to $p(z|x)$, variational inference aims to maximize not the marginal log-probability $\log p(x)$ but rather an expected lower bound on it (the ELBO):

$$\mathcal{L}(x) \triangleq \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)], \tag{3.1}$$

where we have amortized the variational distribution by making it a function of x. The variational autoencoder [Kingma and Welling, 2014; D.J. Rezende, 2014] uses two separate deterministic feedforward neural networks to model $q(z|x)$ and $p(x|z)$, called the recognition (or encoder) model and the generative model, respectively. Specification of the prior $p(z)$ completes the model.

In contrast to undirected generative models [Dayan et al., 1995; Hinton and Salakhutdinov, 2006], sampling from the latent space requires only one feedforward 'sweep.' Additionally, as opposed to stochastic neural networks [Neal, 1992; Mnih and Gregor, 2014; Mnih and Rezende, 2016] the simple parametric form of $q(z|x)$ enables the use of the 'reparameterization trick' [Kingma and Welling, 2014; D.J. Rezende, 2014], which expresses latent samples as functions of parameters of the encoder, often greatly reducing the variance of the training gradients.

The ELBO can be rewritten as

$$\begin{aligned}\mathcal{L}(x) &= \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)] \\ &= \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{\mathrm{KL}}(q(z|x)||p(z)),\end{aligned} \tag{3.2}$$

where we can interpret the first term as encouraging the recognition model to provide latent samples to generative model which will give rise to reconstructions that match the data, while the second term functions as a complexity penalty.

3.3 Prior Choice

Until recently, the prior has received little attention, potentially because from a modeling perspective, a sufficiently complex decoder $p(x|z)$ should be able to transform a base distribution such as the standard Gaussian into a potentially highly complicated marginal distribution over the observed space (see e.g., [Goodfellow et al., 2014]). However, from Equation (3.2), we see that the ELBO is regularized by the encoding distribution's deviation from the prior, so even if a basic prior can produce good samples with the right generative model, if the encoder cannot find a good latent representation, the model will be poor.

3.3.1 Aggregate priors

A key observation by [Hoffman and Johnson, 2016] was that the ELBO averaged over the training set $\{x_n\}_{n=1}^N$ can be rewritten as

$$\mathcal{L}(\theta,\phi) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{q(z|x_n)}[\log p(x_n|z)] - (\log N - \mathbb{E}_{q(z)}[\mathbb{H}[q(n|z)]]) - \mathrm{KL}(q(z)||p(z)) \tag{3.3}$$

$$= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{q(z|x_n)}[\log p(x_n|z)] - \mathrm{MI}[n,z] - \mathrm{KL}(q(z)||p(z)), \tag{3.4}$$

where $\mathbb{H}[\cdot]$ is entropy, $q(n|x) = \frac{q(z|x_n)p_{\mathrm{data}}(x_n)}{q(z)} = \frac{q(z|x_n)}{\sum_{i=1}^N q(z|x_i)}$, and $\mathrm{MI}[\cdot,\cdot]$ is mutual information.

We see from Equation (3.4) that the ELBO consists of a data-dependent reconstruction term

which is regularized by the average mutual information between a data point and its encoding as well as the KL divergence between the marginal variational distribution $q(z)$ and the prior.

Since $p(z)$ appears in the ELBO only though the penalty $KL(q(z)||p(z))$, the optimal prior is $q(z)$. Since $q(z) = \mathbb{E}_{p(x)}[q(z|x)]$, this distribution is often called the aggregate posterior.

[Tomczak and Welling, 2018] approximate this posterior by taking an average over the variational distribution conditioned on a set of pseudoinputs $\{u_k\}_{k=1}^K$ that are optimized:

$$p_{\text{vamp}}(z) = \frac{1}{K}\sum_{k=1}^{K} q(z|u_k). \tag{3.5}$$

Thus, in addition to mapping data points to the latent space, the encoding model must be able to map the pseudoinputs such that their average density in latent space sufficiently matches $q(z)$. This extra constraint may give rise to less precise data encodings; for example, the encoder may need to give pseudoinputs a higher variance in latent space to sufficiently cover $q(z)$, and the increase in the effective image and preimage of the encoder may come at the expense of less precise data encoding.

Instead of using pseudoinputs, [Molchanov et al., 2019] model $p(z)$ semi-implicitly [Yin and Zhou, 2018] as an expectation over a set of stochastic tractable distributions

$$\hat{p}_{\text{SI}}(z) = \frac{1}{K}\sum_{k=1}^{K} q(z|x_{\mathcal{I}_k}), \tag{3.6}$$

where $\mathcal{I}_k \sim \text{Unif}\{1, 2, ..., N\}$.

Due to the convexity of $KL(q(z|x), p(z))$ [Cover and Thomas, 1991], from Equation (3.2), the ELBO evaluated under $\hat{p}(z)$ will be a lower bound on the ELBO under $p(z)$.

Under this model, we do not have to map pseudoinputs in addition to real data, but there is less flexibility in the form that $\hat{p}(z)$ (and therefore $p(z)$) can take, since it relies entirely on the encoded real data. Since $\hat{p}(z)$ must be sufficiently close to $q(z)$, this is a constraint on the encoder;

for example, the encoding model may lose precision in encoding the data points in order for the latent space to be concentrated enough for a random draw of K modes to sufficiently cover it.

3.3.2 Decoupling

In order to keep the advantages of leveraging the encoding model for the prior but lessen the potential disadvantages of doing so, we partially decouple the prior from the encoder to give it flexibility to find a better ELBO, either through a better encoding model that isn't heavily penalized by the KL term or by reducing the KL term while keeping a precise encoding.

We can rewrite the latent distribution as

$$q(z|x_k) \equiv q(z; \phi(x_k)), \tag{3.7}$$

where $\phi(\cdot)$ is the encoder's mapping from data to latent parameters, e.g., $\phi(x) = \{\mu(x), \sigma^2(x)\}$ for a Gaussian encoder. Decoupling is as simple as modifying the parameterization of the aggregate prior in a way that is independent of the encoder. For example, if we wanted to decouple the VampPrior, we would have

$$p_{\text{vamp}+\Delta}(z) = \frac{1}{K}\sum_{k=1}^{K} q(z; \phi(u_k) + \Delta\phi(u_k)), \tag{3.8}$$

where $\Delta\phi(\cdot)$ is any function that maps from the observed space to parameter space. Similarly, we have

$$p_{\text{SI}+\Delta}(z) = \frac{1}{K}\sum_{k=1}^{K} q(z; \phi(x_{\mathcal{I}_k}) + \Delta\phi(x_{\mathcal{I}_k})) \tag{3.9}$$

for the semi-implicit prior. This general approach is shown in Figure 3.1.

Clearly, a change in parameterization is just one way to decouple the prior from the encoder. In the most general sense, $\Delta\phi(x)$ could be any functional mapping $\{q(z; \phi(x_k))\}_{k=1}^K$ to some density $p_\Delta(z)$. However, there are some reasons to decouple via parameter change within the same family of densities. The unique minimizer of Equation (3.3) with respect to $p(z)$ is

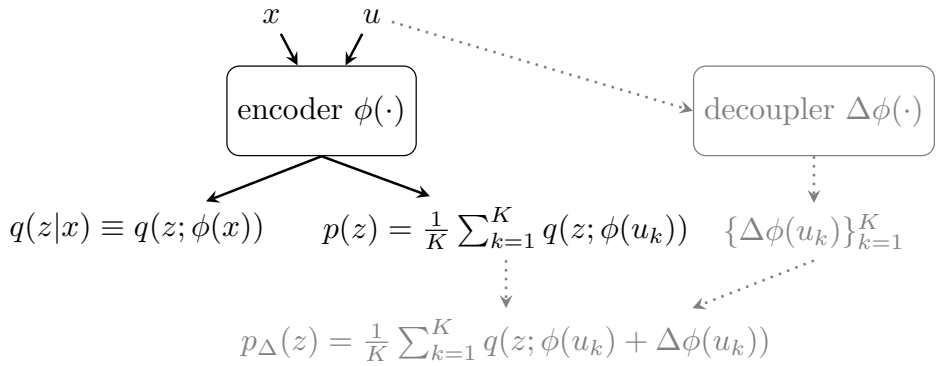


Figure 3.1: Proposed aggregate decoupling model. The vanilla aggregate prior is shown in black and is unchanged; u can represent either pseudoinputs or random data subsamples. Decoupling via the delta function/network is shown in gray and dotted lines.

$p^*(z) = \mathbb{E}_{p_{\text{true}}(x)}[q(z;\phi(x))]$. It may be more accurate to estimate $p^*(z)$ by leveraging knowledge of the encoder model and keeping the integrand in the expectation of the same form. However, we acknowledge there are feasible scenarios in general in which given some samples $\{x_k\}_{k=1}^K$, $\hat{p}(z) = \frac{1}{K}\sum_{k=1}^{K} q(z;\phi(x_k)+\Delta\phi_k)$ may be an inaccurate estimate of $p^*(z)$, for any $\{\Delta\phi_k\}_{k=1}^K$.

We tried a more general decoupling that included learning a weighted combination of densities in addition to the parameter changes, but this performed worse. This is not particularly surprising, as more complicated decoupling models leverage the encoder less. Indeed, [Molchanov et al., 2019] found that a data-independent hierarchical semi-implicit prior performed worse than a random average of the latent encoded distributions in Equation (3.9). Additionally, the parameter change formulation we used is attractive considering its connection with non-parameteric density estimation.

3.3.3 Connection with kernel density estimation

Consider the distribution $p^*(z) = \int q(z;\phi(x))p_{\text{true}}(x)dx$. If we receive samples $\{x_k\}_{k=1}^K$ with corresponding inferred latent variables $\{z_k\}$, we can use a kernel density estimator [Wasserman,

2006] with bandwidth h to give

$$\hat{p}_{\text{KDE}}(z) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(z|z_k, h). \tag{3.10}$$

If we Rao-Blackwellize [Casella and Berger, 2001] by taking the expectation over the latent variables, we get

$$\mathbb{E}_{q(z|x)}[\hat{p}_{\text{KDE}}(z)] = \frac{1}{K}\sum_{k=1}^{K} q(z|x_n) \circledast \mathcal{N}(z|0, h), \tag{3.11}$$

where $\circledast$ indicates convolution. If the encoder is Gaussian with $q(z|x) = \mathcal{N}(z|\mu(x_k), \sigma^2(x_k))$, the right hand side of Equation (3.11) becomes

$$\hat{p}_{\text{KDE-RB}}(z) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(z|\mu(x_k), \sigma^2(x_k) + h). \tag{3.12}$$

Unlike standard kernel density estimation, the ELBO provides a training signal, enabling a more sophisticated model than Equation (3.12) can provide. Specifically, we generalize h to be a function of x (and parameterize it in log space to enable 'negative bandwidths' but an overall non-negative variance for the density) and introduce a kernel biasing function $\Delta\mu(x)$. Renaming $h \triangleq \Delta\sigma^2(x)$ recovers our decoupling model.

Interpreting the prior as kernel density estimation on the latent space opens up a large space of models to explore. A prior that biases the model towards inductive biases, especially on the latent representation could be created by choosing the right basis functions. For example, a good latent space representation often entails that density is not too concentrated at particular points, especially those corresponding to encoded training data points. A common heuristic to check for a good latent space representation is to linearly interpolate between two encoded data points in the latent space and verify that their outputs from the generative model change relatively smoothly and look relatively reasonable (see, e.g., [Ulyanov et al., 2018; Bojanowski et al., 2018; Kingma and Dhariwal, 2018]). A kernel function consisting of two (randomly or otherwise specified) connected

encoded datapoints whose kernel is the distance from a point to the line segment could encourage the model towards such a representation. We implement this prior in Appendix C but do not see improvement. This kernel function could be generalized from the distance from a line segment to the distance from a $d < \dim(z)$ dimensional hyperplane connecting d encoded points.

3.4 Experiments

In all experiments, we followed the architecture for the non-hierarchical VAE in [Tomczak and Welling, 2018]. For the encoding and decoding models, we used a two layer neural network with 300 hidden units per layer with a gated linear unit non-linearity [Dauphin et al., 2017]. The full delta network had the same architecture as the encoder.

For optimization, we used Adam [Kingma and Ba, 2015] with a learning rate of 10^{-4}, gradient norms clipped to unity, and a patience of 50 epochs. We use a linear KL annealing [Bowman et al., 2016] schedule during the first 100 training epochs, as in [Tomczak and Welling, 2018]. All simulations were done in Tensorflow [Abadi et al., 2015].

Unless otherwise stated, we used the same K as reported in [Tomczak and Welling, 2018] and [Molchanov et al., 2019]; no hyperparameters were tuned for our model. For the implicit models, we found that selecting data points uniform randomly without replacement performed better than independently.

For data requiring stochastic binarization, we binarized such that the expected value of each pixel's random value equaled its non-binarized value on $[0, 1]$ as in [Salakhutdinov and Murray, 2008]. Since stochastic binarization can differ across studies, we trained and tested all models on the dynamic datasets and report the maximal log-likelihood across our simulations and [Tomczak and Welling, 2018] and [Molchanov et al., 2019] to make the comparison as generous as possible.

We first tested to see what effect decoupling would have on model performance on three

datasets: static MNIST [Larochelle and Murray, 2011], dynamic MNIST [Salakhutdinov and Murray, 2008], and OMNIGLOT [Lake et al., 2015]. We quantify performance with test log-likelihoods estimated using importance sampling [Burda et al., 2016] with 5000 samples, as in [Tomczak and Welling, 2018]. We report lower bounds on the implicit models' log-likelihoods, since they were evaluated using $\hat{p}(z)$ rather than $p(z)$.

We show our main result in Table 3.1. We found that while decoupling with VampPrior gives incremental improvement, decoupling the SI prior improves its performance from the worst performing model to the best performing model, as measured by the average test log-likelihood. Such a dramatic improvement is at first surprising; however, SI+Δ's connection with kernel density estimation discussed in Section 3.3.3 provides an potential explanation.

In situations in which one is worried about extra computational overhead, adding a delta network on par with the encoding model may be undesirable. We first emphasize that the delta network is generic to the encoder architecture and can be much simpler. To test how sophisticated of a decoupling is needed to improve performance, we tested performance on a 'linear' decoupling network, using only an affine transformation to learn mappings from data space to latent parameter space. We found that linearly decoupling often performed almost as well as decoupling using the same architecture as the encoder (see Table 3.1).

To test each model's robustness to hyperparameter choice, we evaluated performance as a function of K, shown in Figure 3.2. Not only does SI+Δ reach the highest ELBO, but it is quite robust to changes in K, especially higher values of K.

Next, we wanted to see how decoupling improved performance by decomposing the ELBO into the quantities in Equation (3.3), shown in Figure 3.3. Decoupling improved the reconstruction term in both models (Figure 3.3a) as expected.

Perhaps most striking about our results in Figure 3.3 is how high the marginal KL term in

Table 3.1: Test log-likelihoods on three data sets.

Model	Static MNIST	Dynamic MNIST	OMNIGLOT
standard	-88.56	-76.61	-104.15
VampPrior	-85.57	-74.49	-101.85
VampPrior+Δ (linear)	-85.25	-74.75	-100.83
VampPrior+Δ	-85.24	-73.77	-101.63
SI	≥ -89.25	≥ -79.74	≥ -104.66
SI +Δ (linear)	≥ -85.06	≥ -73.64	≥ -100.86
SI +Δ	$\geq \mathbf{-84.91}$	$\geq \mathbf{-73.36}$	$\geq \mathbf{-99.53}$

Figure 3.3b remained even for large values of K. Moreover, while decoupling also improved the $KL(q(z)||p(z))$ term for the SI prior, it increased the term under the VampPrior (Figure 3.3b). This result further illustrates how intertwined the encoder and prior are, especially in aggregate models: the prior does not just influence the marginal KL term, and a good (quantified by the ELBO) prior may have a higher marginal KL than other priors if it enables high enough quality reconstructions.

Consistent with [Hoffman and Johnson, 2016], we found the mutual information term of the ELBO remained relatively constant and close to its maximum of $\log N$.

Finally, we wanted to see if the better encoding model decoupling improved the latent representation. VAEs often suffer having a fraction of latent variables whose encoding is uninformative of the conditioned datapoint, often called 'posterior collapse' or 'mode collapse' ([Burda et al., 2016], [van den Oord et al., 2017], [Dieng et al., 2019]). In Figure 3.4, we compared the number of active units in the latent representation as quantified in [Burda et al., 2016] across models as a

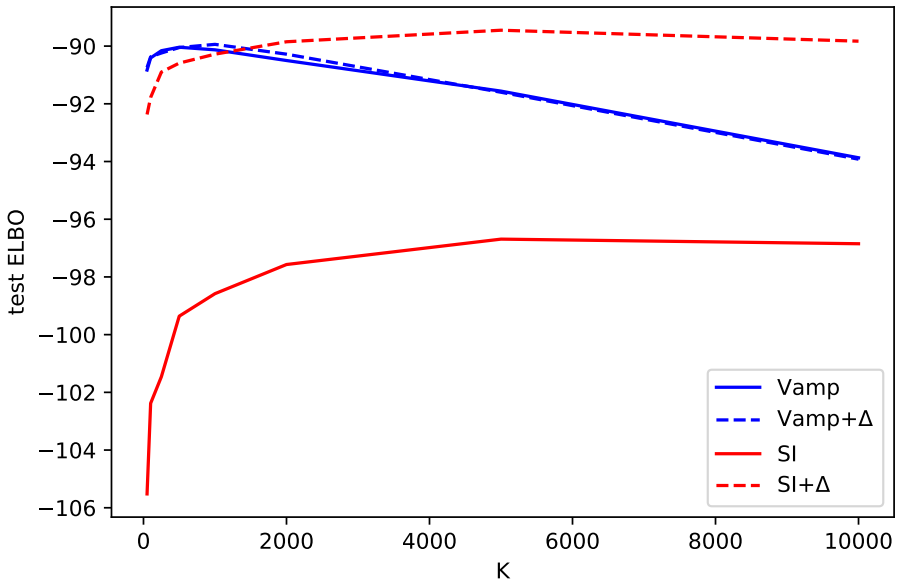


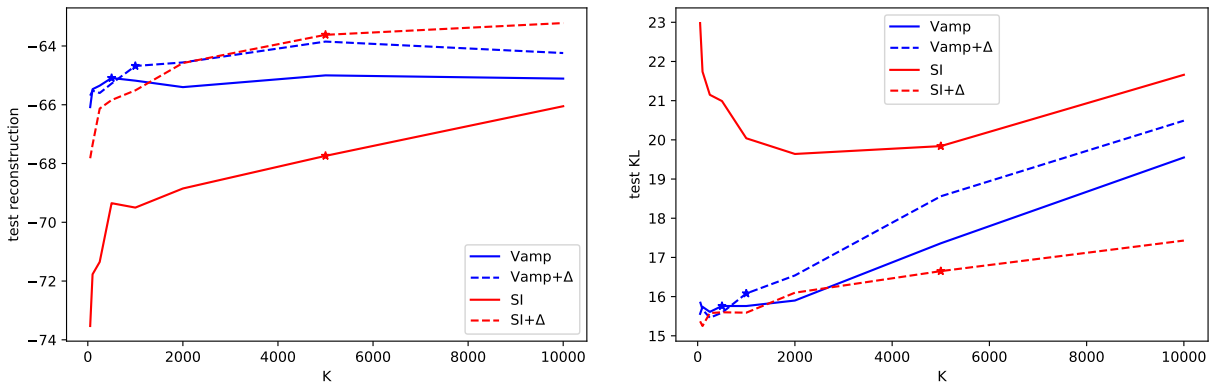
Figure 3.2: Test ELBO terms as a function of K (static MNIST).

function of K using static MNIST as the dataset. We found that decoupling increased the fraction of active units for the VampPrior for all K and increased the number of active units for $K \geq 2000$, well within the regime of best choice of K for SI.

3.5 Conclusion

In this paper, we have generalized the aggregate posterior model of the prior and decoupled it from the encoder model. We have shown decoupling improves the latent representation, and the semi-implicit decoupled prior achieves superior performance over existing aggregate methods on three datasets. For the sake of comparison and simplicity, our prior model has been used in the context of relatively simple feedforward networks; however, it can be easily plugged into more sophisticated encoding and decoding models such as those in [Rezende and Mohamed, 2015], [Sønderby et al., 2016], [Gulrajani et al., 2017], [Chen et al., 2017].

We have shown how the decoupled semi-implicit model can be viewed as an extension of kernel



(a) (b)

Figure 3.3: ELBO terms as a function of K (static MNIST): (a) test reconstruction log-probability, (b) test $KL(q(z)||p(z))$. Asterisks indicate the values of K which maximize the test ELBO for each model in (a) and (b).

density estimation in latent space, opening up new potential avenues for future modeling.

Our work has illustrated not only the importance of the prior in VAEs, but that an aggregate estimate is not the end of the story, thus reiterating the importance of the nuances of the push and pull between data-driven modeling and generalization.

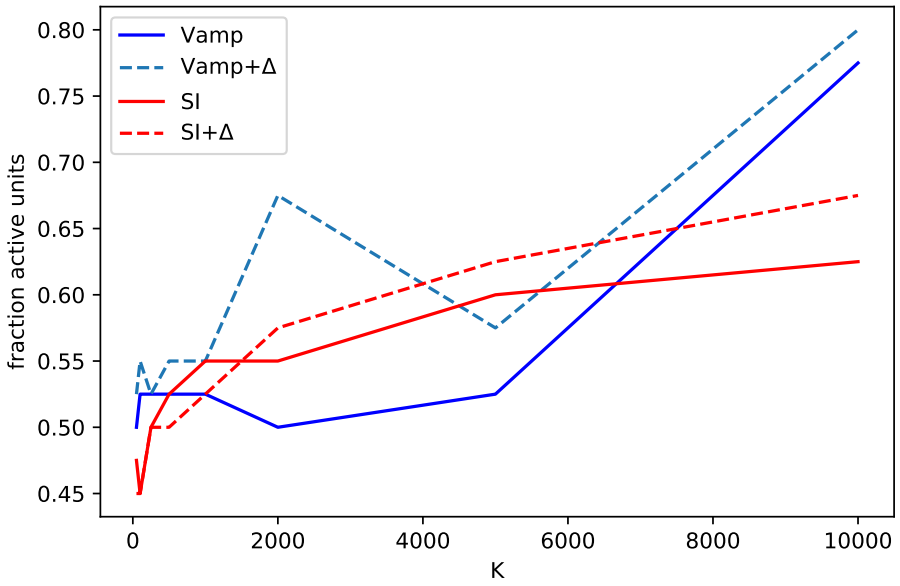


Figure 3.4: Fraction of active latent units for each model as a function of K.

Chapter 4

ELBO amputation: an initialization scheme for variational autoencoders

Variational autoencoders have become ubiquitous in unsupervised learning and deep generative modeling, giving importance to analysis of the finer details and idiosyncrasies of the method. Using a standard architecture, we focus on the behavior of the loss function gradient at the beginning of training and argue that as the latent space dimensionality increases, the data-dependent gradient decays to zero, leaving only the regularizing KL term. We optimize the model with respect to this data-independent gradient estimate with a simple modification to the recognition model's weight initialization which does not add any computational overhead and does not depend on any of the assumptions of the data. Using a sequential variational autoencoder as an example, we show that in addition to speeding up training, models using our initialization converge significantly faster.

4.1 Introduction

Central to the canonical variational autoencoder is the reparameterization trick [Kingma and Welling, 2014; D.J. Rezende, 2014], which rewrites the latent distribution as a function of a random variable drawn from a parameterless base distribution and the latent distribution's parameters, en-

abling differentiation of the latent variable with respect to the variational parameters, which allows the loss function gradient estimate to leverage more information from the variational parameters, thereby reducing the noise of the estimate.

Subsequent efforts have been made to further reduce the noise in the ELBO gradient using control variates [Miller et al., 2017; Geffner and Domke, 2018; Roeder et al., 2017]. Many of these methods rely on the log-derivative trick (REINFORCE) [Williams, 1992], which is simply the identity $\nabla_\phi \mathbb{E}_{q_\phi(z)}[\log q_\phi(z)] = 0$ which follows from the chain rule. The log-derivative trick itself enables derivation of a black-box estimator of the variational gradient [Ranganath et al., 2014] that depends only on function evaluations of the generative and variational models. Since the model can be treated as a black box, the estimator can be used in more general settings than the reparameterization trick, but it suffers from higher variance, although this variance can be reduced using Rao-Blackwellization and control variates [Ranganath et al., 2014].

In this paper, we interpret the black-box form of the variational gradient as a data-dependent cross-covariance minus a data-indepdent KL-divergence complexity penalty gradient. We show that a common network architecture consisting of compositions of affine transformations and ReLU non-linearities [Glorot et al., 2011] with weights initialized using a typical variance-preserving method [Glorot and Bengio, 2010; He et al., 2015] will produce a cross-covariance between the latent variables and the generative parameters whose distribution approaches a point mass at 0 as the latent space dimensionality increases. We use this information to motivation a weight initialization scheme in which the cross-covariance term in the training gradient is ignored and the objective function corresponding to the remaining gradient term is optimized, enabling training to begin at a better initial configuration. We apply our intialization procedure to training a sequential autoencoder [Gregor et al., 2016] and show the training improvements from the initialization persist long into training.

4.2 ELBO gradients

Our starting point is the ELBO:

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)]. \tag{4.1}$$

The gradient of $\mathcal{L}(x)$ with respect to the generative parameters θ is straightforward as the linearity of the gradient enables it to move inside the expectation:

$$\nabla_\theta \mathcal{L}(x) = \mathbb{E}_{q(z|x)}[\nabla_\theta \log p(x,z)]. \tag{4.2}$$

The gradient with respect to the variational parameters is not as simple, as the expectation is with respect to the variation distribution. The key insight from the reparameterization trick [Kingma and Welling, 2014; D.J. Rezende, 2014] is to express $\mathbb{E}_{q(z|x)}[f(z)]$ as an expectation over a parameterless 'base' distribution $q_0(\epsilon)$ and the latent variable $z \sim q(z|x)$ as a function of this base distribution $z = g(\epsilon;\theta)$. Once the expectation doesn't depend on ϕ, the gradient can be moved inside the integral:

$$\nabla_\phi \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)] = \nabla_\phi \mathbb{E}_{q_0(\epsilon)}[\log p(x, g(\epsilon;\phi)) - \log q(g(\epsilon;\phi)|x)] \tag{4.3}$$

$$= \mathbb{E}_{q_0(\epsilon)}[\nabla_\phi(\log p(x, g(\epsilon;\phi)) - \log q(g(\epsilon;\phi)|x))] \tag{4.4}$$

$$= \mathbb{E}_{q_0(\epsilon)}\left[\frac{dg(\epsilon;\phi)}{d\phi}^T \nabla_z(\log p(x,z) - \log q(z|x))\right]. \tag{4.5}$$

The reparameterization trick requires that z can be represented as a differentiable function of some parameterless distribution. However, methods that enable one to relax these assumptions are an increasing area of research (e.g., [Ruiz et al., 2016; Jang et al., 2017; Tucker et al., 2017; Figurnov et al., 2016]).

We can express the gradients in an alternate form by using the log-derivative trick [Williams, 1992], which uses the identity

$$E_{q_\phi(z)}[\nabla_\phi \log q_\phi(z))] = 0, \tag{4.6}$$

where we've made q's parameterization ϕ explicit, to give us

$$\nabla_\phi \mathcal{L}(x) = \mathbb{E}_{q_\phi(z|x)}[\nabla_\phi \log q_\phi(z|x)(\log p(x,z) - \log q_\phi(z|x))]. \tag{4.7}$$

(See [Ranganath et al., 2014] for full details on the derivation.)

4.2.1 Cross-covariance interpretation of gradient

For the sake of simplicity, here we set $p(z) = \mathcal{N}(0, I)$. If $q(z|x) = \mathcal{N}(\mu, 1/\lambda)$, substituting in $\{\mu, \lambda\}$ in for ϕ gives us

$$\nabla_\mu \mathcal{L}(x) = \lambda \mathbb{E}_{q(z|x)}[(z-\mu)\log p(x|z)] - \mu \tag{4.8}$$

$$\nabla_\lambda \mathcal{L}(x) = \frac{1}{2}\mathbb{E}_{q(z|x)}[(1/\lambda - (z-\mu)^2)\log p(x|z)] + \frac{1}{2}(1/\lambda^2 - 1/\lambda). \tag{4.9}$$

If we look at the gradient with respect to μ, we see that μ is pushed in a direction that is proportional to z's cross-covariance with $\log p(x|z)$ and pulled back towards μ by the KL term to match the prior. We see similar behavior for λ's gradient, but with respect to its reciprocal $1/\lambda$. That the gradients break down into penalized cross-covariances makes sense intuitively, since gradients can only represent linear dependence, which is expressed through covariance.

Intuitively, it seems reasonable to expect that a z that is generated by a randomly configured recognition model would be uncorrelated with a log-likelihood that is generated by an independently randomly configured generative model. If $\log p(x|z)$ has a low cross-covariance with z (or $(z-\mu)^2$), then the gradient will reduce to $-\mu$ (or $.5(1/\lambda^2 - 1/\lambda)$). We note that the log-likelihood is a deterministic function of z, so a non-linear measure of dependence like mutual information is maximal.

We will show in Appendix B that if the generative network is a series of affine transformations and ReLUs, under a typical variance-preserving weight initialization scheme [Glorot and Bengio, 2010; He et al., 2015], as the dimensionality of z increases, the cross-covariance between z and the

parameters output by the generative model will go to 0. By 'variance-preserving,' we refer to typical weight initialization schemes in which the variance of the weights for an input scales inversely with the dimensionality of the inputs to keep the variance of the outputs roughly the same as that of the inputs. To see how this works, given an input $x \in \mathbb{R}^d$, we can calculate the variance of the dot product with w, with $w_i \sim \mathcal{N}(0, 1/d)$:

$$\begin{aligned}
\text{Var}\left[\sum_{i=1}^{d} w_i x_i\right] &= \sum_{i=1}^{d} \text{Var}[w_i x_i] \\
&= \sum_{i=1}^{d} \mathbb{E}[w_i^2]\mathbb{E}[x_i^2] - \mathbb{E}[w_i]^2\mathbb{E}[x_i]^2 \\
&= \frac{1}{d}\sum_{i=1}^{d} \text{Var}[x_i] \\
&= \overline{\text{Var}[x_i]_{i=1}^{d}}
\end{aligned}$$

We note our result does not prove that the cross-covariance terms in (4.8) and (4.9) are equal to zero, since $\log p(x|z)$ is a non-linear function of the variational parameters, but our result can at least serve as motivation for empirical testing of our initialization.

Our initialization modification is extremely simple: once the recognition model is initialized (the particular initialization scheme does not matter as long as it is variance-preserving, e.g., [Glorot and Bengio, 2010; He et al., 2015]), we set every entry in the matrix that determines the linear mapping from the last hidden layer to each parameter μ and $\log \lambda$ to 0, such that $q(z|x) = \mathcal{N}(0, I) = p(z)$ for all x, eliminating the KL term.

Another interpretation of this modification is that right after initialization, we can pre-train the model using the data-independent KL-gradient. With our zero initialization scheme, we are finding a solution to $\forall x \, \nabla_\phi \mathcal{L}(x) = 0$. If we wanted to avoid a fully zero intialization, we could find the singular value decomposition of the empirical covariance of the last layer's activations and project a random initialization of the weights onto a linear subspace orthogonal to the first few

singular vectors. Although we have found that only relatively few singular vectors explain much of the covariance, performance using pure zeros has performed better than sampling from this orthogonal subspace.

4.2.2 Potential concerns: code collapse and symmetry

One may argue that by giving such an initialization, the latent space is not encouraged to be a meaningful representation, since the latent space is initially simply a degenerate point mass. In fact, it is possible that this is the worst initialization possible, since we are intentionally creating the greatest amount of latent space 'code collapse.' However, any initial latent representation that results from random initialization will induce a random topography onto the latent space that will then need to be undone or transformed to create a mapping that maps similar data points together. Initially mapping onto the origin circumvents having to perform this unmapping.

Another potential issue is that initializing weights to 0 might induce a symmetry in the architecture such that individual units cannot be differentiated from one another, leading to each unit in a layer becoming identical, thus reducing the layer to a single computational unit. Intuitively, this does not happen here because the symmetry among the latent variables is broken by the random mapping in the generative network from latent space to image space. Thus, $\frac{\partial \mathcal{L}(x)}{\partial \mu}$ contains different entries in general, and since $\mu(x) = Wh(x)+b$, where $h(x)$ is the last hidden layer, $\frac{\partial \mu}{\partial W}$ is a function of $h(x)$ and is unaffected by the weight matrix's value. Therefore, $\frac{\partial \mathcal{L}(x)}{\partial W} = \frac{\partial \mu}{\partial W}^T \frac{\partial \mathcal{L}(x)}{\partial \mu}$ does not take any special or degenerate form under our initialization. An identical argument holds for λ.

4.2.3 Numerical simulation

In Section 4.2, we argued that as the dimensionality of the latent space grows, the generative term of the ELBO, which can be interpreted as the covariance between the latent space and the

generative log-likelihood, will go to 0. We show this empirically in Fig. 4.1 using a single hidden layer densely connected neural network for both the generative model and the recognition model. The dimensionality of the latent space is 100, and the data used is the MNIST dataset restricted to the central 8x8 square (for memory reasons, since we take one million samples from the latent space to get extremely accurate MCMC gradients). The dimensionality of the hidden layer is 10 for memory reasons, but we had qualitatively similar performance with a dimensionality of 100. We see that as the model begins to train, this similarity begins to break down, as the generative model loses its initial random configuration as it trains (and its log-likelihood becomes correlated with the latent space). This doesn't happen until the ELBO has already made considerable improvements, suggesting that training the generative model gives rise to the slow and steady improvements in the ELBO.

4.2.4 Application to sequential autoencoder

In standard applications of variational autoencoders, zero initialization may not give rise to noticeable improvements, as the latent space dimensionality must be sufficiently high for the cross-covariance to be dominated by the KL gradient. However, one scenario in which the latent space dimensionality is high is when using sequential autoencoders [Gregor et al., 2016].

Sequential autoencoders represent data in a temporal sequence, rather than the output of a single pass of e.g., a feedforward network. The advantage of this approach is that data can be represented in a sequence of latent encodings, rather than a single encoding that must capture all the information about the data. The following equations illustrate a basic sequential autoencoder

model for image data:

$$\hat{x}_t = x - \sigma(c_{t-1}) \tag{4.10}$$

$$r_t = \text{read}(x_t, \hat{x}_t, h_{t-1}^{\text{dec}}) \tag{4.11}$$

$$h_t^{\text{enc}} = \text{RNN}^{\text{enc}}(h_{t-1}^{\text{enc}}, [r_t, h_{t-1}^{\text{dec}}]) \tag{4.12}$$

$$z_t = q(z_t|h_t^{\text{enc}}) \tag{4.13}$$

$$h_t^{\text{dec}} = \text{RNN}^{\text{dec}}(h_{t-1}^{\text{dec}}, z_t) \tag{4.14}$$

$$c_t = c_{t-1} + \text{write}(h_t^{\text{dec}}), \tag{4.15}$$

where $\hat{x}_t$ is the error image $x_t - \sigma(c_t)$, $[\cdot, \cdot]$ denotes concatenation of vectors, $\sigma(\cdot)$ is the sigmoid function, and c_t is the 'canvas' that becomes the generative model's output as $\sigma(c_T)$. The read() and write() functions are general and can be used with or without attention.

Under this framework, the KL penalties for each timestep $KL(q(z_t|x)||p(z))$ are additive: the latent space is effectively a concatenation of each timestep's latent encoding. This property makes such a model a good candidate for zero initialization.

To test this, we trained a sequential autoencoder with 10 time steps and a latent and hidden dimensionality of 100 on MNIST. We used Glorot initialization [Glorot and Bengio, 2010] on all models, but there was no qualitative difference in results when using He initialization [He et al., 2015]. Figure 4.2a shows a substantial improvement from zero initialization over normal initialization. We also note the effective dimensionality of 1000 is much smaller than that of the models in the original sequential autoencoder paper [Gregor et al., 2016], suggesting even better improvements could be seen when training the larger model.

Annealing the KL term [Bowman et al., 2016; Sønderby et al., 2016] is a heuristic used to encourage a VAE to learn a meaningful latent representation. At the start of training, models with sophisticated decoders (e.g., [van den Oord et al., 2016a; van den Oord et al., 2016b; Kingma et al.,

2016]) do not need an informative latent space to improve reconstruction error; consequently, the latent space can remain uninformative for the entirety of training while only the decoder model learns the temporal structure of the observed data. Since this method alleviates the impact of the KL term on the gradient, it should be compared to our method.

Figure 4.2b shows that zero initialization performs better than all the (linear) annealing schedules we tried. Interestingly, the annealing schedules' ELBOs appear to approach that of the zero initialization as the annealing schedule is lengthened.

4.3 Discussion

We have shown that a simple weight initialization modification can have long lasting improvements in training certain types of variational autoencoders. For the sake of simplicity, we used a standard normal prior, but our procedure could be amended to more sophisticated priors, such as a mixture of Gaussians or a VampPrior [Tomczak and Welling, 2018], for example, by modeling the parameters of $q(z|x)$ as those of $p(z)$ plus the output of a neural network and zero-initializing the last linear layer of the network.

Better-tuned weight initialization has been a significant contributing factor in enabling training networks to use simple first order optimization rather than greedy layerwise pretraining [Hinton and Salakhutdinov, 2006; Bengio et al., 2007] or Hessian-free optimization [Martens, 2010]. A small, seemingly trivial multiplicative factor of $\sqrt{2}$ [He et al., 2015] enables the training of very deep networks previously untrainable. We anticipate as deep learning progresses details that appear to be minutiae at first glance will make the difference in achieving state-of-the-art performance.

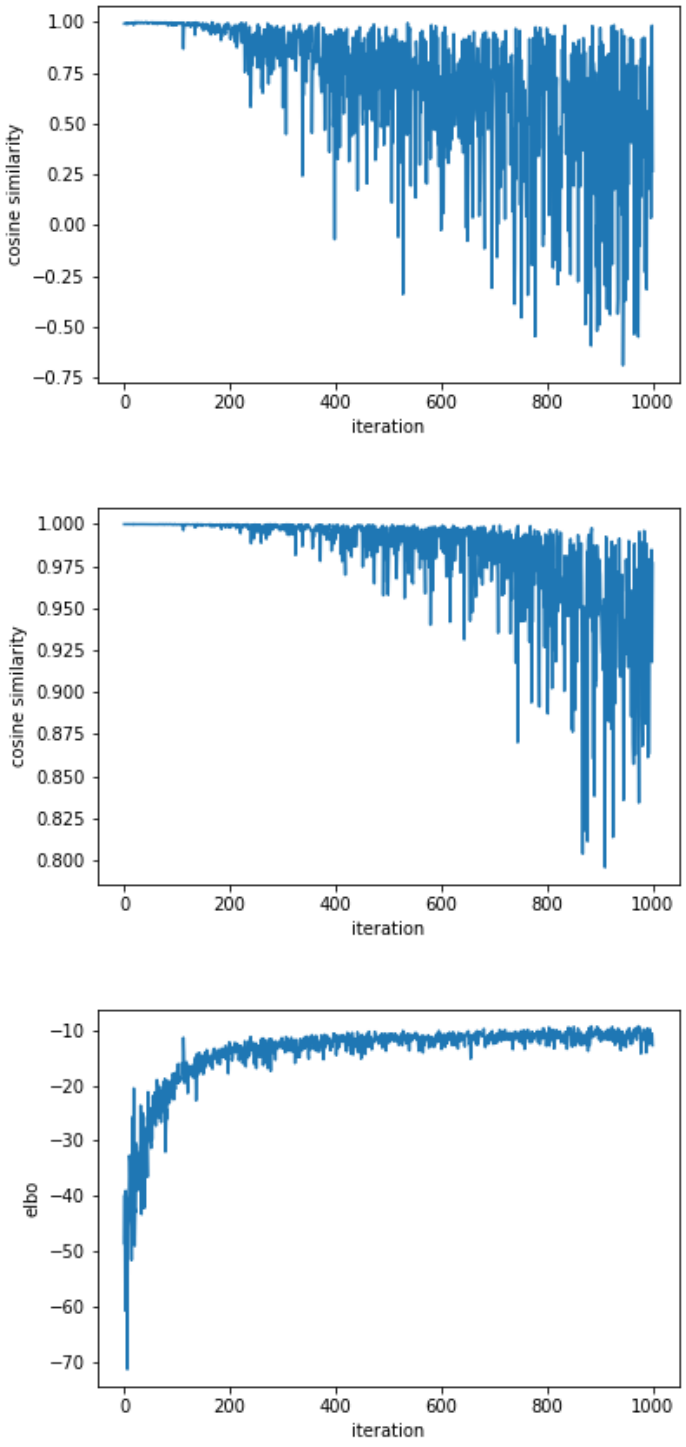
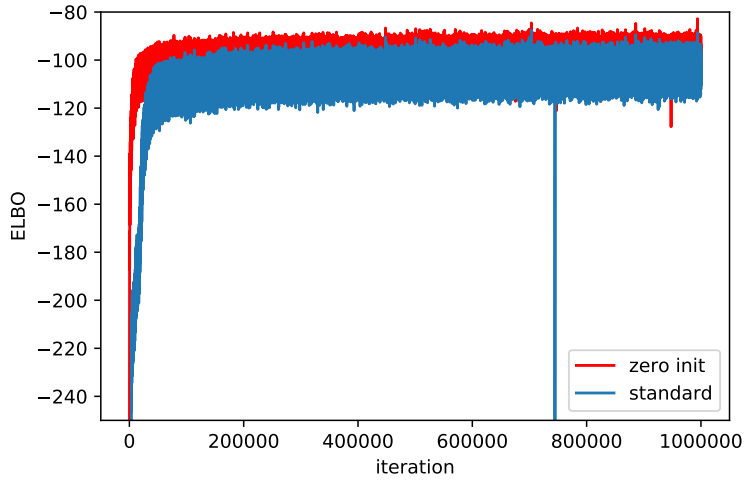
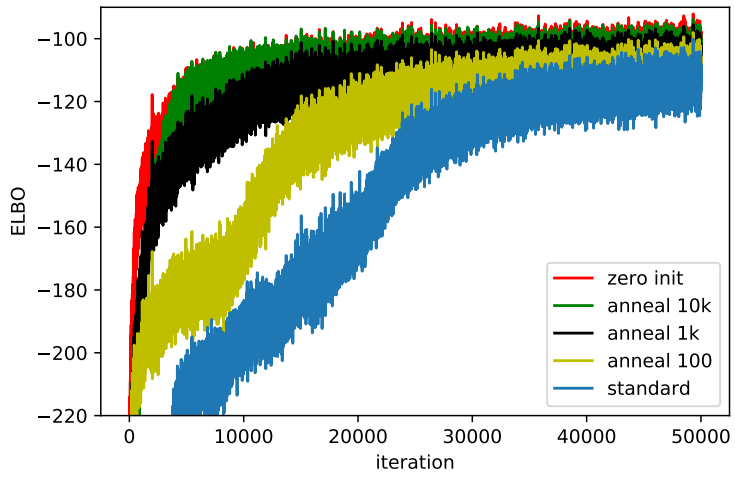


Figure 4.1: Closed-form KL gradients dominate during the beginning of training. Top: Cosine similarity between MCMC estimated μ gradient for and closed-form μ KL gradient during training. Middle: Same for λ gradients. Bottom: ELBO during training.



(a)



(b)

Figure 4.2: Training a sequential autoencoder on MNIST. Comparison of test ELBOs during training between zero initialization and (a) standard initialization (b) various KL annealing schedules. The length of annealing in iterations for each schedule is written after 'anneal' in the legend.

Chapter 5

Conclusion

"All speech is demand. Every demand a request for love." — Jacques Lacan

In this thesis, we have improved partition function estimation to improve training of undirected graphical models, generalized a class of prior models in variational autoencoders leading to better performance and latent representation, and modified random initialization schemes for autoencoders to improve training convergence speed. Along the way, we have drawn parallels between several existing methods for partition function estimation, pointed out an implicit constraint in a variant of variational inference, and exploited symmetries in randomly initialized neural networks. Science often requires us to get well within the weeds to make progress, but it is equally important after all is said and done to take a step back and do some introspection.

Science exists in the interplay between theory and practice; a scientist attempts to understand the world through hypothesizing models and subsequently testing their validity by collecting data and comparing it to the model's predictions. This 'scientific method' we learned in grade school seems simple at first–almost a tautology–but details get in the way. Even if we ignore the looming question of the the validity of inductive reasoning in general, there are still nuances that thwart us.

How do we come up with good hypotheses that are informed by past experiments but unbiased with respect to data used to test the model's predictions? The interplay is not without gaps; there is no complete bridge between theoretical models and real (even fake) data, giving way to subjectivity, interpretation, storytelling. Statistics is very good at saying no but sometimes we cannot help but whisper in our heads, '*Yes....*' Thus, despite our best efforts, science remains a study of ourselves.

Deep learning's massive popularity makes it a particularly interesting case in the sociology of science. The breadth and depth of progress deep learning has made in machine learning is difficult to overstate, so much so that at times it can be hard to believe Yoshua Bengio's response to being asked what is the biggest misconception about deep learning: that it isn't magic. In much of science, the devil is in the details: small seemingly trivial changes to protocol (in machine learning's case, the model or training/testing procedure) give rise to qualitatively different results, which in turn drive subsequent directions for theory and experiments. Hyperparameter 'twiddling' to achieve state-of-the-art results is a big problem in machine learning, but an even bigger problem arises when the practitioner is unaware of the degree of arbitrariness of the study and all the other hidden knobs left untouched. In a blog post, Andrej Karpathy writes of accidentally leaving a model training over winter break only to come back to it giving state-of-the-art performance. Of course, neural networks themselves are to a degree arbitrary, a subset of models in the class of universal function approximators. A cynic could argue that the biggest factor in the success of neural networks has been the name.

The contradictions in our attempt to separate out signal and noise is poetic, but upon further inspection not limited to science: in any framework, there is an arbitrary starting point, assumptions, relations. And there is always more structure to be uncovered (or invented), more theory to be done.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

[Alemi et al., 2017] Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., A.Saurous, R., and Murphy, K. (2017). An information-theoretic analysis of deep latent-variable models. arXiv:711.00464.

[Atchade and Liu, 2010] Atchade, Y. and Liu, J. (2010). The Wang-Landau algorithm in general state spaces: Applications and convergence analysis. *Statistica Sinica*.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*.

[Bennett, 1976] Bennett, C. (1976). Efficient estimation of free energy differences from Monte Carlo data. *Journal of Computational Physics*.

[Beskos et al., 2013] Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*.

[Bojanowski et al., 2018] Bojanowski, P., Joulin, A., Paz, D. L., and Szlam, A. (2018). Optimizing the latent space of generative networks. *International Conference on Machine Learning*.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *CompSTAT*.

[Bowman et al., 2016] Bowman, S., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. *CoNLL*.

[Burda et al., 2015] Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Accurate and conservative estimates of MRF log-likelihood using reverse annealing. *International Confernce on Artificial Intelligence and Statistics*.

[Burda et al., 2016] Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance-weighted autoencoders. *International Conference on Learning Representations*.

[Calderhead and Girolami, 2009] Calderhead, B. and Girolami, M. (2009). Estimating Bayes factors via thermodynamic integration and population MCMC. *Computational Statistics & Data Analysis*, 53(12):4028–4045.

[Carlson et al., 2015] Carlson, D., Collins, E., Hsieh, Y.-P., Carin, L., and Cevher, V. (2015). Preconditioned spectral descent for deep learning. In *Advances in Neural Information Processing Systems*, pages 2953–2961.

[Casella and Berger, 2001] Casella, G. and Berger, R. (2001). *Statistical Inference*. Cengage Learning.

[Cattell, 1952] Cattell, R. (1952). *Factor analysis*. Harper.

[Chen et al., 2018] Chen, R., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *Advances in Neural Information Processing Systems*.

[Chen et al., 2017] Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017). Variational lossy autoencoder. *International Conference on Learning Representations*.

[Cover and Thomas, 1991] Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley-Interscience.

[Dauphin et al., 2017] Dauphin, Y., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. *International Conference on Machine Learning*.

[Dayan et al., 1995] Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*.

[Dellaportas and Kontoyiannis, 2012] Dellaportas, P. and Kontoyiannis, I. (2012). Control variates for estimation based on reversible Markov Chain Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(1):133–161.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*.

[Desjardins et al., 2011] Desjardins, G., Bengio, Y., and Courville, A. (2011). On tracking the partition function. In *Advances in Neural Information Processing Systems*, pages 2501–2509.

[Desjardins et al., 2010] Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. (2010). Tempered Markov Chain Monte Carlo for training of Restricted Boltzmann Machines. In *International Conference on Artificial Intelligence and Statistics*, pages 145–152.

[Dieng et al., 2019] Dieng, A., Kim, Y., Rush, A., and Blei, D. (2019). Avoiding latent variable collapse with generative skip models. *International Confernce on Artificial Intelligence and Statistics*.

[Dinh et al., 2017] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. *International Conference on Learning Representations*.

[D.J. Rezende, 2014] D.J. Rezende, S. Mohamed, D. W. (2014). Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*.

[Figurnov et al., 2016] Figurnov, M., Mohamed, S., and Mnih, A. (2016). Implicit reparameterization gradients. *Advances in Neural Information Processing Systems*.

[Friel et al., 2014] Friel, N., Hurn, M., and Wyse, J. (2014). Improving power posterior estimation of statistical evidence. *Statistics and Computing*.

[Friel and Wyse, 2012] Friel, N. and Wyse, J. (2012). Estimating the evidence–a review. *Statistica Neerlandica*, 66(3):288–308.

[Gabrie et al., 2015] Gabrie, M., Tramel, E., and Krzakala, F. (2015). Training Restricted Blotzmann Machines via the Thouless-Anderson-Palmer free energy. In *Advances in Neural Information Processing Systems*, pages 640–648.

[Geffner and Domke, 2018] Geffner, T. and Domke, J. (2018). Using large ensembles of control variates for variational inference. *Advances in Neural Information Processing Systems*.

[Gelman and Meng, 1998] Gelman, A. and Meng, X.-L. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Geyer and Thompson, 1995] Geyer, C. and Thompson, E. (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431):909–920.

[Geyer, 1994] Geyer, C. J. (1994). Estimating normalizing constants and reweighting mixtures.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *International Confernce on Artificial Intelligence and Statistics*.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. *International Confernce on Artificial Intelligence and Statistics*.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio., Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*.

[Gregor et al., 2016] Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2016). DRAW: A recurrent neural network for image generation. *Advances in Neural Information Processing Systems*.

[Grosse et al., 2013] Grosse, R., Maddison, C., and Salakhutdinov, R. (2013). Annealing between distributions by averaging moments. In *Advances in Neural Information Processing Systems*, pages 2769–2777.

[Grosse and Salakhudinov, 2015] Grosse, R. and Salakhudinov, R. (2015). Scaling up natural gradient by sparsely factorizing the inverse Fisher matrix. In *Proceedings of the 32nd International Conference on Machine Learning (International Conference on Machine Learning-15)*, pages 2304–2313.

[Gulrajani et al., 2017] Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A., Visin, F., Vazquez, D., and Courville, A. (2017). PixelVAE: a latent variable model for natural images. *International Conference on Learning Representations*.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *ICCV*.

[Hinton, 2002] Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*.

[Hinton and Salakhutdinov, 2006] Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313.

[Hoffman and Johnson, 2016] Hoffman, M. D. and Johnson, M. (2016). ELBO surgery: yet another way to carve up the variational evidence lower boundn. *Advances in Neural Information Processing Systems Workshops*.

[Huszar, 2017] Huszar, F. (2017). Variational Inference using Implicit Distributions. arXiv:1702.08235.

[Im et al., 2015] Im, D., Buchman, E., and Taylor, G. (2015). Understanding minimum probability flow for RBMs under various kinds of dynamics. *International Conference on Learning Representations Workshop Track*.

[Jang et al., 2017] Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with Gumbel-softmax. *International Conference on Learning Representations*.

[Jordan et al., 1999] Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: a method for stochastic optimization. *International Conference on Learning Representations*.

[Kingma and Dhariwal, 2018] Kingma, D. and Dhariwal, P. (2018). Glow: generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*.

[Kingma et al., 2016] Kingma, D., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*.

[Kingma and Welling, 2014] Kingma, D. and Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Machine Learning*.

[Lake et al., 2015] Lake, B., Salakhutdinov, R., and J.B.Tenenbaum (2015). Human-level concept learning through probabilistic program induction. *Science*.

[Larochelle and Murray, 2011] Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. *International Confernce on Artificial Intelligence and Statistics*.

[Li et al., 2004] Li, Y., Protopopescu, V., and Gorin, A. (2004). Accelerated simulated tempering. *Physics Letters A*.

[Liu et al., 2015] Liu, Q., Peng, J., Ihler, A., and III, J. F. (2015). Estimating the partition function by discriminance sampling. *Conference on Uncertainty and Artificial Intelligence*.

[Makhzani et al., 2016] Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. (2016). Adversarial autoencoders. *International Conference on Learning Representations*.

[Marin and Robert, 2009] Marin, J.-M. and Robert, C. (2009). Importance sampling methods for Bayesian discrimination between embedded models. *arXiv preprint arXiv:0910.2325*.

[Marinari and Parisi, 1992] Marinari, E. and Parisi, G. (1992). Simulated tempering: a new monte carlo scheme. *EPL (Europhysics Letters)*, 19(6):451.

[Martens, 2010] Martens, J. (2010). Deep learning via Hessian-free optimization. *International Conference on Machine Learning*.

[Mescheder et al., 2017] Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes:unifying variational autoencoders and generative adversarial networks. *International Conference on Machine Learning*.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *J Chem Phys*.

[Miller et al., 2017] Miller, A., Fotit, N., DAmour, A., and Adams, R. (2017). Reducing reparameterization gradient variance. *Advances in Neural Information Processing Systems*.

[Mnih and Gregor, 2014] Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. *International Conference on Machine Learning*.

[Mnih and Rezende, 2016] Mnih, A. and Rezende, D. (2016). Variational inference for monte carlo objectives. *International Conference on Machine Learning*.

[Molchanov et al., 2019] Molchanov, D., Kharitonov, V., Sobolev, A., and Vetrov, D. (2019). Doubly semi-implicit variational inference. *International Conference on Machine Learning*.

[Mortimer et al., 2009] Mortimer, D., Feldner, J., Vaughan, T., Vetter, I., Pujic, Z., Rosoff, W., Burrage, K., Dayan, P., Richards, L., and Goodhill, G. (2009). A Bayesian model predicts the response of axons to molecular gradients. *PNAS*.

[Nalisnick and Smyth, 2017] Nalisnick, E. and Smyth, P. (2017). Stick-breaking variational autoencoders. *International Conference on Learning Representations*.

[Neal, 2001] Neal, R. (2001). Annealed importance sampling. *Statistics and Computing*.

[Neal, 2005] Neal, R. (2005). Estimating ratios of normalizing constants using linked importance sampling. *arXiv preprint math/0511216*.

[Neal, 2011] Neal, R. (2011). *Handbook of Markov Chain Monte Carlo*, chapter MCMC using Hamiltonian dynamics. Chapman & Hall / CRC Press.

[Neal, 1992] Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113.

[Oates et al., 2015] Oates, C., Papamarkou, T., and Girolami, M. (2015). The controlled thermodynamic integral for Bayesian model evidence evaluation. *Journal of the American Statistical Association*.

[Ranganath et al., 2016] Ranganath, R., Altosaar, J., Tran, D., and Blei, D. (2016). Operator variational inference. *Advances in Neural Information Processing Systems*.

[Ranganath et al., 2014] Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. *International Confernce on Artificial Intelligence and Statistics*.

[Rezende and Mohamed, 2015] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. *International Conference on Machine Learning*.

[Robert and Casella, 2013] Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

[Roeder et al., 2017] Roeder, G., Wu, Y., and Duvenaud, D. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *Advances in Neural Information Processing Systems*.

[Ruiz et al., 2016] Ruiz, F., Titsias, M., and Blei, D. (2016). The generalized reparameterization gradient. *Advances in Neural Information Processing Systems*.

[Salakhutdinov, 2009] Salakhutdinov, R. (2009). Learning in markov random fields using tempered transitions. In *Advances in neural information processing systems*, pages 1598–1606.

[Salakhutdinov, 2010] Salakhutdinov, R. (2010). Learning deep Boltzmann machines using adaptive MCMC. In *Proceedings of the 27th International Conference on Machine Learning (International Conference on Machine Learning-10)*, pages 943–950.

[Salakhutdinov and Murray, 2008] Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of Deep Belief Networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 872–879.

[Serban et al., 2017] Serban, I., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio., Y. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. *AAAI Conference on Artificial Intelligence*.

[Shi et al., 2018] Shi, J., Sun, S., and Zhu, J. (2018). Kernel implicit variational inference. *International Conference on Learning Representations*.

[Shirts and Chodera, 2008] Shirts, M. and Chodera, J. (2008). Statistically optimal analysis of samples from multiple equilibrium states. *The Journal of Chemical Physics*, 129(12):124105.

[Smolensky, 1986] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document.

[Sohl-Dickstein et al., 2010] Sohl-Dickstein, J., Battaglino, P., and DeWeese, M. (2010). Minimum probability flow learning. *International Conference on Machine Learning*.

[Sønderby et al., 2016] Sønderby, C., Raiko, T., Maaløe, L., Sønderby, S., and Winther, O. (2016). Ladder variational autoencoders. *Advances in Neural Information Processing Systems*.

[Tan et al., 2016] Tan, A., Xia, J., Zhang, B., and Levy, R. (2016). Locally weighted histogram analysis and stochastic solution for large-scale multi-state free energy estimation. *The Journal of Chemical Physics*, 144(3):034107.

[Tan, 2016] Tan, Z. (2016). Optimally adjusted mixture sampling and locally weighted histogram analysis. *Journal of Computational and Graphical Statistics*.

[Tan et al., 2012] Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R. (2012). Theory of binless multi-state free energy estimation with applications to protein-ligand binding. *The Journal of Chemical Physics*, 136(14):144102.

[Tenenbaum et al., 2011] Tenenbaum, J., Kemp, C., Griffiths, T., and Goodman, N. (2011). How to grow a mind: Statistics,structure, and abstraction. *Science*.

[Tieleman and Hinton, 2009] Tieleman, T. and Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1033–1040. ACM.

[Tomczak and Welling, 2018] Tomczak, J. and Welling, M. (2018). VAE with a VampPrior. *International Confernce on Artificial Intelligence and Statistics*.

[Tran et al., 2017] Tran, D., Ranganath, R., and Blei, D. (2017). Hierarchical implicit models and likelihood-free variational inference. *Advances in Neural Information Processing Systems*.

[Tran et al., 2016] Tran, D., Ranganath, R., and Blei, D. M. (2016). The variational Gaussian process. *International Conference on Learning Representations*.

[Tucker et al., 2017] Tucker, G., Mnih, A., Maddison, C., Lawson, D., and Sohl-Dickstein, J. (2017). REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*.

[Ulyanov et al., 2018] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018). It takes (only) two: Adversarial generator-encoder networks. *AAAI Conference on Artificial Intelligence*.

[van de Meent et al., 2014] van de Meent, J.-W., Paige, B., and Wood, F. (2014). Tempering by subsampling. *arXiv preprint arXiv:1401.7145*.

[van den Berg et al., 2018] van den Berg, R., Hasenclever, L., Tomczak, J., and Welling, M. (2018). Sylvester normalizing flows for variational inference. *Conference on Uncertainty and Artificial Intelligence*.

[van den Oord et al., 2016a] van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016a). Pixel recurrent neural networks. *International Conference on Machine Learning*.

[van den Oord et al., 2016b] van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016b). Conditional image generation with PixelCNN decoders. *Advances in Neural Information Processing Systems*.

[van den Oord et al., 2017] van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. *Advances in Neural Information Processing Systems*.

[Vehtari et al., 2014] Vehtari, A., Gelman, A., Sivula, T., Jylanki, P., Tran, D., Sahai, S., Blomstedt, P., Cunningham, J., Schiminovich, D., and Robert, C. (2014). Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data. arXiv:1412.4869.

[Vyshemirsky and Girolami, 2008] Vyshemirsky, V. and Girolami, M. (2008). Bayesian ranking of biochemical system models. *Bioinformatics*, 24(6):833–839.

[Wang and Landau, 2001] Wang, F. and Landau, D. P. (2001). Efficient multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters E*.

[Wasserman, 2006] Wasserman, L. (2006). *All of Nonparametric Statistics*. Springer.

[Williams, 1992] Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

[X.-L.Meng and Wong, 1996] X.-L.Meng and Wong, W. (1996). Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, 6(4):831–860.

[Xu et al., 2014] Xu, M., Lakshminarayanan, B., Teh, Y., Zhu, J., and Zhang, B. (2014). Distributed Bayesian posterior sampling via moment sharing. *Advances in Neural Information Processing Systems*.

[Yin and Zhou, 2018] Yin, M. and Zhou, M. (2018). Semi-implicit variational inference. *International Conference on Machine Learning*.

[Z. et al., 2015] Z., R. G., Ghahramani, and Adams, R. (2015). Sandwiching the marginal likelihood using bidirectional Monte Carlo. *arXiv preprint arXiv:1511.02543*.

Appendix A

Chapter 2 Appendix

A.1 Estimating $q(\beta_k)$ from a transition matrix

Instead of estimating $q(\beta_k)$ by Rao-Blackwellizing via c_k in (2.9), it is possible to estimate $q(\beta_k)$ from the stationary distribution of a transition matrix. The key idea here is that the transition matrix accounts for the sampling structure used in MCMC algorithms, whereas c_k is derived using i.i.d. samples. Suppose that we have a Gibbs sampler sequence $x_1 \to \beta_1 \to x_2 \cdots \to \beta_N$. For a Gibbs sampler, this sequence can be collapsed to Markov transitions over β_k with

$$\begin{aligned} p(\beta_{n+1} = \beta_k | \beta_n = \beta_j), & \\ &= \sum_x p(\beta_{n+1} = \beta_k | x) p(x | \beta_n = \beta_j), \\ &= P_{jk}. \end{aligned}$$

The top eigenvector of P gives the stationary distribution over β_k, which is $q(\beta_k)$. We briefly mention two strategies to estimate this transition matrix. First, this matrix can simply be estimated with empirical samples, with

$$P_{jk} \propto \sum 1_{\{\beta_n+1=\beta_k, \beta_n=\beta_j\}},$$

where $1_{\{.\}}$ is the identity function. Then $q(\beta_k)$ is estimated from the top eigenvector. We denote this strategy Stationary Distribution (SD). A second approach is to Rao-Blackwellize over the samples, where

$$P_{jk} \propto \sum p(\beta_n + 1 = \beta_k | x_n) 1_{\{\beta_n = \beta_j\}}.$$

We denote this strategy as Rao-Blackwellized Stationary Distribution (RSD).

The major drawback of this approach is that it is rare to have exact Gibbs samples over $p(x|\beta)$, but instead we have a transition operation $T(x_n|\beta, x_{n-1})$. In this case, it is unclear whether this approach is useful. We note that in simple cases, such as a RBM with 10 hidden nodes, RSD can sizably reduce the RMSE over RTS, as shown in Figure A.1(Left). However, in more complicated cases when the assumption that we have a Gibbs sampler over $p(x|\beta)$ breaks down, there is essentially no change between RTS and RSD, as shown in a 200 hidden node RBM in Figure A.1 (Right). Our efforts to correct the transition matrix for the transition operator instead of a Gibbs sampler did not yield performance improvements.

A.2 Adaptive HMC for tempering

Here we consider sampling from a continuous distribution using Hamiltonian Monte Carlo (HMC) [Neal, 2011]. Briefly, HMC simulates Hamiltonian dynamics as a proposal distribution for Metropolis-Hastings (MH) sampling. In general, one cannot simulate exact Hamiltonian dynamics, so usually one uses the leapfrog algorithm, a first order discrete integration scheme which maintains the time-reversibility and volume preservation properties of Hamiltonian dynamics.

[Li et al., 2004] found using different step sizes improved sampling various multimodal distributions using random walk Metropolis proposal distributions. However, under their scheme, besides step sizes being monotonically decreasing in β, it is unclear how to set these step sizes. Additionally, in target distributions that are high-dimensional or have highly correlated variables,

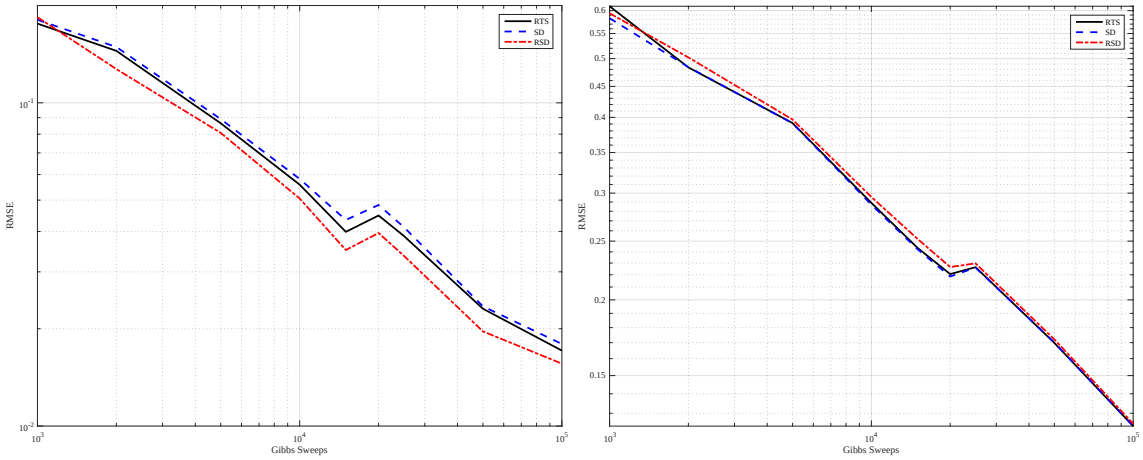


Figure A.1: An illustration of the effect of estimating the stationary distribution from the transition matrix. Both plots show the RMSE on RBMs averaged over 20 repeats. Experimental procedure is the same as the main text. (Left) RTS, TM, and RTM compared on a 784-10 RBM. Because the latent dimensionality is small, mixing is very effective and accounting for the transition matrix improves performance consistently by about 10%. (Right) For an 784-200 RBM, the approximation as a Markov transition is inaccurate, and we observe no performance improvements.

random walk Metropolis will work badly.

For most distributions of interest, as β decreases, $p(x|\beta)$ becomes flatter; thus, for HMC, we can expect the MH acceptance probability to decrease as a function of β, enabling us to take larger jumps in the target distribution when the temperature is high. As the stepsize of the leapfrog integrator gets smaller, the linear approximation of the solution to the continuous differential equations becomes more accurate, and the MH acceptance probability increases (for an infinitely small stepsize, the simulation is exact, and under Hamiltonian dynamics, the acceptance probability is 1). Thus, $p(\text{accept}|\epsilon)$ decreases with ϵ. Putting this idea together, we model $p(\text{accept}|\beta, \epsilon)$ as a logistic function for each $\beta \in \{0 = \beta_1, ..., \beta_J = 1\}$

$$\text{logit}(p(\text{accept}|\beta, \epsilon)) = w_0^{(j)} + w_1^{(j)}\epsilon \tag{A.1}$$

Given some data $\{(\beta^{(i)}, s^{(i)}, y^{(i)})\}_{i=1,...,N}$ ($y^{(i)} = 1$ if proposed sample i was accepted, and $y^{(i)} = 0$

if it was rejected), we find

$$\begin{aligned} \max_{\{w^{(j)}\}} \quad & \sum_{j=1}^{J} h(w^{(j)}) \\ \text{s.t.} \quad & w_1^{(j)} \leq 0 \\ & g(\beta_j, \epsilon) \leq g(\beta_{j-1}, \epsilon) \ \ \forall\, \epsilon \end{aligned} \tag{A.2}$$

where

$$h(w^{(j)}) = \sum_{i:\beta^{(i)}=\beta_j} y^{(i)} \log(g(\beta^{(i)}, \epsilon^{(i)})) + (1 - y^{(i)}) \log(1 - g(\beta^{(i)}, \epsilon^{(i)}))$$

and

$$g(\beta_j, \epsilon) = p(\text{accept}|\beta_j, \epsilon) = \frac{1}{1 + \exp(-(w_0^{(j)} + w_1^{(j)}\epsilon))}$$

The last constraint can be satisfied by enforcing $g(\beta_j, \epsilon_{\min}) \leq g(\beta_{j-1}, \epsilon_{\min})$ and $g(\beta_j, \epsilon_{\max}) \leq g(\beta_{j-1}, \epsilon_{\max})$, as doing so will ensure $g(\beta_j, \epsilon) \leq g(\beta_{j-1}, \epsilon)$ for all $\epsilon \in [\epsilon_{\min}, \epsilon_{\max}]$. Before solving (A.2), we first run chains at fixed $\beta = 0$ and $\beta = 1$, running a basic stochastic optimization method to adapt each stepsize until the acceptance rate is close to the target acceptance rate, which we take to be 0.651, which is suggested by [Beskos et al., 2013]. We take these stepsizes to be $\epsilon_{\max}$ and $\epsilon_{\min}$, respectively. Once we have approximated $p(\text{accept}|\beta, \epsilon)$, choosing the appropriate proposal distribution given β is simple:

$$\hat{\epsilon}_{\text{opt}}(\beta_j) = \frac{\text{logit}(\text{p(acc)}) - w_0^{(j)}}{w_1^{(j)}}$$

If $\hat{\epsilon}_{\text{opt}}$ is outside $[\epsilon_{\min}, \epsilon_{\max}]$, we project it into the interval.

Consider a target distribution of a mixture of two 10-dimensional Gaussians, each having a covariance of $0.5I$ separated in the first dimension by 5. Our prior distribution for the interpolating scheme is a zero mean Gaussian with covariance $30I$. The prior was chosen by looking at a one-dimensional projection of the target distribution and picking a zero-mean prior whose variance, σ^2

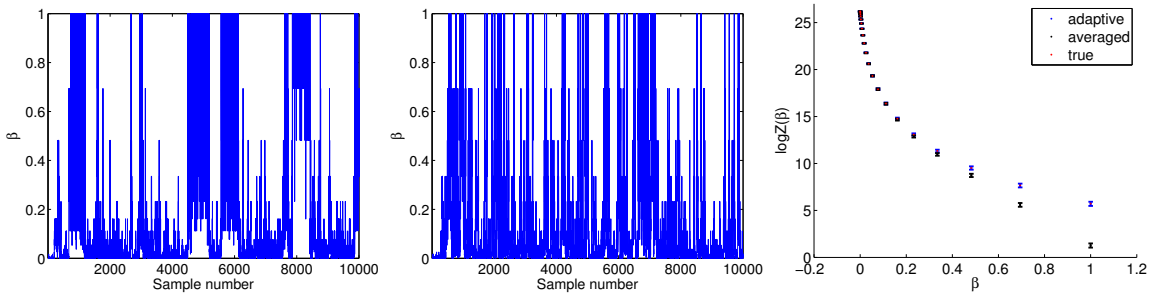


Figure A.2: (Left) Mixing in β under the fixed step size. (Center) Mixing in β under the adaptive scheme. (Right) Partition function estimates under the fixed step size and adaptive scheme after 10000 samples. Mixing in β using a fixed step size is visibly slower than mixing using the adaptive step size, which is reflected by the error in the partition function estimate.

adequately covered both of the modes. The variance of the multidimensional prior was taken to be $\sigma^2 I$, and the mean to be 0. Our prior on temperatures was taken to be uniform. We compare the adaptive method above to simulation with a fixed step size, which is determined by averaging all of the step sizes, in an effort to pick the optimal fixed step size. The below figures show an improvement over the fixed step size in mixing and partition function estimation using our adaptive scheme.

We obtained similar improvements using random walk Metropolis by varying the covariance of an isotropic Gaussian proposal distribution. We note another scheme for discrete binary data may be used, where the number of variables in the target distribution to "flip", as a function of temperature, is a parameter.

A.3 Similarity of RTS and MBAR

We also note that there is a particularly interesting relationship between the the cost function for MBAR and the cost function for RTS. Note that $\mathbb{E}_q[\frac{n_k}{N}]$ is equal to $q(\beta_k)$ for tempered sampling. If the values of $\frac{n_k}{N}$ in (2.21) are replaced by their expectation, the maximizer of (2.21) is equal to

the RTS estimator given in (2.12). To prove this, we first restate the the likelihood of MBAR given in (2.21):

$$L[\mathbf{Z}] = \frac{1}{N}\sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\frac{n_k}{N}\exp(-\log Z_k + \beta_k\Delta_{x_i})\right) + \sum_{k=1}^{N}\frac{n_k}{N}\log Z_k$$

The partial derivative of this likelihood with respect to $\log Z_k$ is given by:

$$\frac{\partial L[\mathbf{Z}]}{\partial \log Z_k} = \frac{n_k}{N} - \frac{1}{N}\sum_{i=1}^{N}\frac{\frac{n_k}{N}\exp(-\log Z_k + \beta_k\Delta_{x_i})}{\sum_{j=1}^{K}\frac{n_j}{N}\exp(-\log Z_j + \beta_j\Delta_{x_i})} \tag{A.3}$$

Replacing $\frac{n_k}{N}$ with its expectation for all k gives

$$\frac{\partial L[\mathbf{Z}]}{\partial \log Z_k} = q(\beta_k) - \frac{1}{N}\sum_{i=1}^{N}\frac{q(\beta_k)\exp(-\log Z_k + \beta_k\Delta_{x_i})}{\sum_{j=1}^{K}q(\beta_j)\exp(-\log Z_j + \beta_j\Delta_{x_i})} \tag{A.4}$$

Noting that $q(\beta_k) \propto Z_k/\hat{Z}_k r_k$, we have

$$\begin{aligned}
\frac{\partial L[\mathbf{Z}]}{\partial \log Z_k} &= q(\beta_k) \\
&\quad - \frac{1}{N}\sum_{i=1}^{N}\frac{\frac{Z_k}{\hat{Z}_k}r_k\exp(-\log Z_k + \beta_k\Delta_{x_i})}{\sum_{j=1}^{K}\frac{Z_j}{\hat{Z}_j}r_j\exp(-\log Z_j + \beta_j\Delta_{x_i})}, \\
&= q(\beta_k) \\
&\quad - \frac{1}{N}\sum_{i=1}^{N}\frac{\exp(-\log \hat{Z}_k + \beta_k\Delta_{x_i})}{\sum_{j=1}^{K}\exp(-\log \hat{Z}_j + \beta_j\Delta_{x_i})}, \\
&= q(\beta_k) - \frac{1}{N}\sum_{i=1}^{N}q(\beta_k|x_i), \\
&= q(\beta_k) - \hat{c}_k\,.
\end{aligned} \tag{A.5}$$

Setting the partial derivative to 0 and substituting the definition of $q(\beta)$ into (A.5) gives a solution of

$$\frac{Z_k/\hat{Z}_k r_k}{\sum_{j=1}^{K} Z_j/\hat{Z}_j r_j} = \hat{c}_k, \tag{A.6}$$

which is identical to the RTS update in (2.12).

Hence, the similarity of MBAR and RTS will depend on how far the empirical counts vary from their expectation.

A.4 RTS and TI-RB Continuous β Equivalence

We want to show the relationship mentioned in 2.3.4, which we repeat here:

$$\begin{aligned}\log\left(\frac{\hat{Z}_K}{Z_1}\right)^{(RTS)} &= \int_0^1 \frac{d}{d\beta}\left(\log \hat{c}_\beta - \log r_\beta + \log \hat{Z}_\beta\right) d\beta, \\ &= \int_0^1 \frac{\sum_i q(\beta|x_i)\Delta_{x_i}}{\sum_j q(\beta|x_j)} d\beta\,.\end{aligned}$$

Note that we can write the statistics c_k as

$$\begin{aligned}c_k &= \sum_{i=1}^{N} q(\beta_k|x_i) \\ &= \sum_{i=1}^{N} \frac{\exp\left(\beta_k \Delta_{x_i} + \log r_k - \log \hat{Z}_k\right)}{\sum_{k'=0}^{K} \exp\left(\beta_{k'} \Delta_{x_i} + \log r_{k'} - \log \hat{Z}_{k'}\right)}\end{aligned}$$

The continuous version of this replaces the index k by β, and

$$\begin{aligned}c_\beta &= \sum_{i=1}^{N} q(\beta|x_i) \\ &= \sum_{i=1}^{N} \frac{\exp\left(\beta \Delta_{x_i} + \log r_\beta - \log \hat{Z}_\beta\right)}{\int_0^1 \exp\left(\alpha \Delta_{x_i} + \log r_\alpha - \log \hat{Z}_\alpha\right) d\alpha}\end{aligned}$$

The continuous form of the RTS estimator can be written as an integral:

$$\begin{aligned}\log \frac{Z_K}{Z_1} &= \left(\log c_\beta - \log r_\beta + \log \hat{Z}_\beta\right)\Big|_{\beta=1} \\ &\quad - \left(\log c_\beta - \log r_\beta + \log \hat{Z}_\beta\right)\Big|_{\beta=0} \\ &= \int_0^1 \frac{d}{d\beta}\left(\log c_\beta - \log r_\beta + \log \hat{Z}_\beta\right) d\beta \end{aligned} \tag{A.7}$$

We first analyze the derivative of c_β, which is

$$\begin{aligned}&\frac{d}{d\beta}\log c_\beta \\ =&\frac{d}{d\beta}\log \sum_{i=1}^{N} \frac{\exp\left(\beta\Delta_{x_i} + \log r_k - \log \hat{Z}_k\right)}{\int_0^1 \exp\left(\alpha\Delta_{x_i} + \log r_\alpha - \log z_\alpha\right) d\alpha} \\ =&\frac{1}{\sum_{i=1}^{N} \frac{\exp\left(\beta\Delta_{x_i} + \log r_\beta - \log \hat{Z}_\beta\right)}{\int_0^1 \exp\left(\alpha\Delta_{x_i} + \log r_\alpha - \log \hat{Z}_\alpha\right) d\alpha}} \\ &\times \sum_{i=1}^{N} \frac{\exp\left(\beta\Delta_{x_i} + \log \frac{r_\beta}{\hat{Z}_\beta}\right) \frac{d}{d\beta}\left(\beta\Delta_{x_i} + \log \frac{r_\beta}{\hat{Z}_\beta}\right)}{\int_0^1 \exp\left(\alpha\Delta_{x_i} + \log r_\alpha - \log \hat{Z}_\alpha\right) d\alpha} \\ =&\sum_i \frac{q(\beta|x_i)\frac{d}{d\beta}\left(\beta\Delta_{x_i} + \log r_\beta - \log \hat{Z}_\beta\right)}{\sum_j q(\beta|x_j)} \\ =&\left[\sum_i \frac{q(\beta|x_i)}{\sum_j q(\beta|x_j)}\Delta_{x_i}\right] + \frac{d}{d\beta}(\log r_\beta - \log \hat{Z}_\beta) \end{aligned} \tag{A.8}$$

The last line follows since $\sum_{i=1}^{N} \frac{q(\beta|x_i)}{\sum_j q(\beta|x_j)} = 1$. The $\frac{d}{d\beta}(\log r_\beta - \log \hat{Z}_\beta)$ term in (A.7) and (A.8) simply cancel.

We stress that while the continuous formulation of RTS and TI-RB are equivalent in the continuous limit, in the discrete case RTS *does not* suffer from discretization error. And we reiterate that RTS is of course limited to the case when samples are generated by the joint tempered distribution $q(x, \beta)$.

Parallels between other methods and Thermodynamic Integration can be drawn as well. As

noted in [Neal, 2005], the log importance weight for AIS can be written as

$$\log w = \sum_{k=2}^{K} (\beta_k - \beta_{k-1}) \Delta_{x_k} \tag{A.9}$$

and thus can be thought of as a Riemann sum approximation to the numerical integral under a particular sampling approach.

Appendix B

Chapter 4 Appendix

In this section, we examine $\text{Cov}[z, \theta(z)] = \mathbb{E}_z[z\theta(z)] - \mathbb{E}_z[z]\mathbb{E}_z[\theta(z)]$, where $\theta(z)$ is a function we model with increasing complexity: first a linear function, then a linear function followed by a ReLU nonlinearity [Glorot et al., 2011], then a collection of linear+ReLU functions corresponding to a single layer of a neural network, and then finally a sequence of layers to make a complete neural network. Such an architecture is currently one of the most commonly used architectures and includes (nonpooling) convolutional networks. We omit biases, as they are typically initialized to 0.

For each function $\theta(z)$, we find an expression for the distribution of the cross-covariance and show that as the dimensionality of the latent space d_z increases, the distribution converges to a point mass at 0. More specificially, the cross-covariance between z and the network output is Gaussian distributed with mean 0 and covariance that scales inversely with the dimensionality of z.

For simplicity, we assume weight initialization draws values i.i.d from $\mathcal{N}(0, \frac{1}{n_{\text{in}}})$, where n_{in} is the dimensionality of the input. This initialization keeps forward-propagated variances the same and is called 'Lecun normal' initialization. He initialization [He et al., 2015] multiplies this value by

$\sqrt{2}$ to account for on average half of ReLU units being activated to 0. Glorot/Xavier initialization [Glorot and Bengio, 2010] uses an inverse variance of $(n_{\text{in}} + n_{\text{out}})/2$ in an attempt to balance forward- and back-propagated variances.

B.1 Linear function

In the linear case, $\theta(z) = w^T z$, giving a cross-covariance of

$$\begin{aligned}\mathbb{E}_z[zw^T z] - \mathbb{E}_z[z]\mathbb{E}_z[w^T z] &= \mathbb{E}[zz^T]w - \mathbb{E}[z]\mathbb{E}[w^T z] \\ &= \Lambda^{-1}w,\end{aligned}$$

which gives us

$$\text{Cov}[z, \theta(z)] \sim \mathcal{N}(0, 1/d_z\Lambda^{-2}). \tag{B.1}$$

B.2 Linear + ReLU function

Consider $\theta(z) = \text{relu}(w^T z)$, which is a basic building block of many neural networks. Recall $\text{relu}(x) \triangleq \max(x, 0)$.

We begin by evaluating $\mathbb{E}_z[\text{relu}(w^T z)]$. We can eliminate the non-linearity inside the expectation by rewriting the integral as over a linear function constrained to the positive halfspace induced by w defined as $H_w^+ \triangleq \{z : w^T z \geq 0\}$:

$$\mathbb{E}_z[\text{relu}(w^T z)] = \int_z \mathbb{I}(w^T z > 0) w^T z p(z) dz \tag{B.2}$$

$$= \int_{H_w^+} w^T z p(z) dz \tag{B.3}$$

$$= w^T \left(\int_{H_w^+} z p(z) dz \right) \tag{B.4}$$

.

Next, we reparameterize z as the transformation of a standard normal random variable $\epsilon \sim \mathcal{N}(0, I)$ as $z = \Lambda^{-1/2}\epsilon + \mu$. The corresponding inequality constraint on ϵ is $\tilde{w}^T\epsilon \geq -w^T\mu$, where $\tilde{w} \triangleq \Lambda^{-1/2}w$, to give

$$\int_{H_w^+} zp(z)dz = \Lambda^{-1/2} \int_{H_{\tilde{w}}^+} \epsilon p(\epsilon)d\epsilon + \mu \tag{B.5}$$

where $H_{\tilde{w}}^+ \triangleq \{\epsilon : \tilde{w}^T\epsilon \geq -\mu^T w\}$.

We can perform a change of variables by rotating the hyperplane that defines our halfspace so that it is orthogonal to ϵ_1's axis and parallel to all others. In other words we can choose an orthonormal matrix M such that $M\tilde{w} = ||\tilde{w}||e_1$, where e_i is the ith unit vector. If we define $\tilde{\epsilon} \triangleq M\epsilon$ then the inequality constraint halfspace becomes

$$\{\epsilon : \tilde{w}^T\epsilon \geq -\mu^T w\} = \{\epsilon : (M\tilde{w})^T(M\epsilon) \geq -\mu^T w\} \tag{B.6}$$

$$= \{\tilde{\epsilon} : (||\tilde{w}||e_1)^T\tilde{\epsilon} \geq -\mu^T w\} \tag{B.7}$$

$$= \{\tilde{\epsilon} : \tilde{\epsilon}_1 \geq -\mu^T w/||\tilde{w}||\} \triangleq H_{M\tilde{w}}^+. \tag{B.8}$$

This rotation enables the Gaussian integral's dimensions to decouple, enabling taking expectations separately and all of the expectations over the real line zeroing out:

$$\begin{aligned}
&\int_{H_{\tilde{w}}^+} \epsilon p(\epsilon)d\epsilon = M^T \int_{H_{M\tilde{w}}^+} \tilde{\epsilon} p(\tilde{\epsilon})d\tilde{\epsilon} \\
&= M^T \left[e_1 \int_{\tilde{\epsilon}_1 \geq -\mu^T w/||\tilde{w}||} \tilde{\epsilon}_1 p(\tilde{\epsilon}_1)d\tilde{\epsilon}_1 + e_2 \int_{-\infty}^{\infty} \tilde{\epsilon}_2 p(\tilde{\epsilon}_2)d\tilde{\epsilon}_2 + \right. \\
&\qquad \left. \cdots + e_{d_z} \int_{-\infty}^{\infty} \tilde{\epsilon}_{d_z} p(\tilde{\epsilon}_{d_z})d\tilde{\epsilon}_{d_z} \right] \\
&= M^T \left[\sqrt{2\pi} \left(1 - \Phi\left(-\frac{\mu^T w}{||\tilde{w}||} \right) \right) \exp\left(\frac{(\mu^T w)^2}{2||\tilde{w}||^2} \right) e_1 + (1 - e_1)0 \right] \\
&= \sqrt{2\pi} \left(1 - \Phi\left(-\frac{\mu^T w}{||\tilde{w}||} \right) \right) \exp\left(\frac{(\mu^T w)^2}{2||\tilde{w}||^2} \right) \frac{\tilde{w}}{||\tilde{w}||} \\
&= \beta\Lambda^{-1/2}w,
\end{aligned} \tag{B.9}$$

where $\beta \triangleq \sqrt{2\pi}\left(1 - \Phi\left(-\alpha\right)\right)\exp\left(1/2\alpha^2\right)$ and $\alpha \triangleq \frac{\mu^T w}{2||\tilde{w}||}$.

By the linearity of the expectation, we get

$$\mathbb{E}_z[z] = \Lambda^{-1/2}\mathbb{E}_\epsilon[\epsilon] + \mu = \beta\Lambda^{-1}w + \mu. \tag{B.10}$$

The second moment is found in the same way:

$$\mathbb{E}_\epsilon[\epsilon\epsilon^T] = I + \frac{(\delta - 1)}{w^T\Lambda^{-1}w}\Lambda^{-1/2}ww^T\Lambda^{-1/2}, \tag{B.11}$$

where $\delta \triangleq \int_{-\mu^T w/w^T\Lambda^{-1}w}^{\infty} x^2 \exp(-1/2x^2)dx \lessapprox 2.5$.

Putting this in terms of z gives

$$\begin{aligned}
\mathbb{E}_z[zz^T] &= \mathbb{E}_\epsilon[(\Lambda^{-1/2}\epsilon + \mu)(\Lambda^{-1/2}\epsilon + \mu)^T] \\
&= \Lambda^{-1/2}\mathbb{E}_\epsilon[\epsilon\epsilon^T]\Lambda^{-1/2} + \beta\Lambda^{-1}w\mu^T + \beta\mu w^T\Lambda^{-1} + \mu\mu^T \\
&= \Lambda^{-1} + \beta\Lambda^{-1}w\mu^T + \beta\mu w^T\Lambda^{-1} + \mu\mu^T
\end{aligned}$$

Putting this together, we have

$$\begin{aligned}
\mathrm{Cov}[z, \mathrm{relu}(w^T z)] &= \mathbb{E}_z[z\mathrm{relu}(w^T z)] - \mathbb{E}_z[z]\mathbb{E}_z[\mathrm{relu}(w^T z)] && \text{(B.12)} \\
&= (1 - \Phi(-\alpha))(\Lambda^{-1} + \beta\Lambda^{-1}w\mu^T)w && \text{(B.13)} \\
&= (1 - \Phi(-\alpha))(\Lambda^{-1}w + \beta\Lambda^{-1}ww^T\mu) && \text{(B.14)}
\end{aligned}$$

This is just the linear case plus a covariance correction factor for w and scaled by the probability of z being in the positive halfspace of w. The second term is small relative to the first one, since the magnitude of each element in ww^T scales inversely with the square of the dimensionality of z, d_z, whereas the magnitude of each element of the first term scales inversely with d_z.

B.3 Linear + ReLU layer

If we have multiple linear units within a layer represented by Wz instead of $w^T z$, the cross-covariance is now a matrix instead of a vector:

$$\begin{aligned}\mathrm{Cov}[z, \mathrm{relu}(Wz)] = \big[&\mathrm{Cov}[z, \mathrm{relu}(W_1^T z)], \mathrm{Cov}[z, \mathrm{relu}(W_2^T z)], \\ &\cdots, \mathrm{Cov}[z, \mathrm{relu}(W_{d_h}^T z)]\big],\end{aligned} \tag{B.15}$$

where d_h is the number of hidden units in the layer.

The parameters of $p(x|z)$ (before non-linearities) are determined by a linear combination of the activations of the last layer

$$\theta = V\mathrm{relu}(Wz), \tag{B.16}$$

where $V_{i,j} \sim \mathcal{N}(0, \frac{1}{d_h} I)$. This transformation gives us

$$\begin{aligned}&\mathbb{E}_z[z\theta_i] - \mathbb{E}_z[z]\mathbb{E}_z[\theta_i] = \\ &\sum_{j=1}^{d_h} V_{ij}(1 - \Phi(-\alpha_j)) \left(\Lambda^{-1} W_j + \beta_j \Lambda^{-1} W_j W_j^T \mu\right),\end{aligned} \tag{B.17}$$

where α and β are now vectors of length d_h.

The linear combination preserves the cross-covariance scaling; that is, the covariance $\mathrm{Cov}[z, \theta_i]$ will be equal to the average covariance across $\mathrm{Cov}[z, \mathrm{relu}(W_j^T z)]_{j=1}^{d_h}$.

B.4 Linear + ReLU network

We are now interested in a multilayer neural network in which the tth layer's activation can be written as

$$z^{(t)} = \text{relu}(W^{(t)} z^{(t-1)}) \tag{B.18}$$

$$= \text{relu}(W^{(t)} \text{relu}(W^{(t-1)} z^{(t-2)})) \tag{B.19}$$

$$= \text{relu}(W^{(t)} \text{relu}(W^{(t-1)} \text{relu}(W^{(t-2)} \cdots W^{(2)} \text{relu}(W^{(1)} z^{(0)}) \cdots))). \tag{B.20}$$

We can rewrite each ReLU unit as

$$\text{relu}(Wz) = \Omega_W^+(z) W z, \tag{B.21}$$

where $\Omega_W^+(z) \triangleq \text{diag}([\mathbb{I}(z \in H_{W_i}^+)]_{i=1}^{d_h})$.

In other words, for a fixed z, the ReLU can be thought of as a linear transformation Wz where certain rows of W are zeroed out depending on z. Thus, z space is broken up into 2^{d_h} partitions corresponding to all the possible binary strings $\Omega_W^+(z)$ can take. Within each partition, the ReLU function reduces to a linear transformation.

Consider the transformation of z through two ReLU layers

$$\tilde{z} = \text{relu}(W \text{relu}(Vz)) \tag{B.22}$$

$$= \text{relu}(W \Omega_V^+(z) V z) \tag{B.23}$$

$$= \text{relu}(\tilde{W} z), \tag{B.24}$$

where $\tilde{W} \triangleq W \Omega_V^+(z) V$.

Looking at the i, jth element in $\tilde{W}$, we have

$$\tilde{W}_{ij} = \sum_{k=1}^{H} W_{ik}\Omega_{kk}V_{kj} \tag{B.25}$$

$$= \sum_{k=1}^{H} \Omega_{kk}u_k, \tag{B.26}$$

where $u_k \triangleq W_{ik}V_{jk}$ comes from a normal product distribution of mean 0 and variance $1/{d_h}^2$. Looking at the variance of $\Omega_{kk}u_k$, we get

$$\mathrm{Var}[\Omega_{kk}u_k] = \mathbb{E}[(\Omega_{kk}u_k)^2] - \mathbb{E}[\Omega_{kk}u_k]^2 \tag{B.27}$$

$$= \mathbb{E}[\Omega_{kk}u_k^2] = \mathbb{E}[\Omega_{kk}]\mathbb{E}[u_k^2] = \frac{1}{2{d_h}^2}. \tag{B.28}$$

If d_h is sufficiently large, the Central Limit Theorem gives us

$$\tilde{W}_{ij} = \sum_{k=1}^{d_h} \Omega_{kk}u_k \tag{B.29}$$

$$\sim \mathcal{N}\left(0, \frac{1}{2d_h}\right). \tag{B.30}$$

If we follow [He et al., 2015] and multiply our initialized weight matrices by $\sqrt{2}$, we get

$$\tilde{z} = \mathrm{relu}(\tilde{W}z),\ \tilde{W}_{ij} \sim \mathcal{N}\left(0, \frac{1}{d_h}\right), \tag{B.31}$$

thus reducing to the single layer ReLU case in Section B.3. By induction, we can reduce any length composition of layers in this way.

Appendix C

Gaussian tube prior

In Chapter 3, after showing that the SI+Δ prior is very similar to modeling the prior as a Gaussian kernel density estimator over the observed latent variables, we suggested using different kernel functions for problems with different inductive biases, etc.

The Gaussian tube prior differs from a typical kernel density estimator in that the distance is from a point z to the line $\overline{z_1 z_2}$, instead of a single point z_1 or z_2. We use a Gaussian kernel over that distance, giving

$$K_h(z; z_1, z_2) \propto \exp\left(-.5 d_h^2(z, \overline{z_1 z_2})\right), \tag{C.1}$$

where $d_h(\cdot, \cdot)$ represents a Mahalanobis distance parameterized by h.

This formulation is motivated by a common test for a good latent representation being that the latent variables along the line segment between two latent points inferred from two data points should have a relatively high density and thus look reasonable, ideally giving rise to a smooth transition between the two latent points.

The pairs in Equation (C.1) can be picked deterministically or stochastically. We found picking pairs of points stochastically with a probability inversely proportional to their distance from each

other worked best. The KDE formulation of the prior is then

$$\hat{p}(z) \triangleq \frac{1}{\tilde{N}} \sum_{k=1}^{\tilde{N}} K_h^{(k)}(z; z_i, z_j). \tag{C.2}$$

We note that this basis function is just one of many that can be constructed; for example, one could model a non-constant density along the line segment or take the distance from a hyperplane for the Gaussian kernel.

C.1 Examining latent space

We created a toy dataset to test the interpretability of a model's latent space. The dataset consists of squares with all pixels value 1 whose size and position change on a background of 0. After the model has been trained by optimizing the ELBO, we take two data points, sample from the latent space $q(z|x)$ for each, and interpolate several latent points between each sampled latent point. We then look at the resulting $p(x|z)$ for each of these latent points.

We choose the two data points to interpolate between to be extremes of one abstract aspect of the dataset. Each data point in the set is determined by two values–size and position–so we look at interpolations across these attributes. The first interpolation fixes the size of the square and changes its position from right to left. The second interpolation fixes the position of the square at a corner and decreases its size from large to small. According to the intuition behind the heuristic, the interpolations should show a smooth transition between the two endpoints.

An explanation for the Gaussian tube not performing significantly better than all the competitor priors can be given in the form of a counterexample. Suppose we had a perfectly smooth latent space that interpolated perfectly. Consider adding an invertible function $f^{-1}(z)$ at the top of the generative model, i.e., the new generative model is now $\tilde{p}(x|z) = p(x|f^{-1}(z))$, and transforming our latent space $z \sim q(z|x)$ into $\tilde{z} \sim f(z)$. The new model is equivalent to the old model, but the

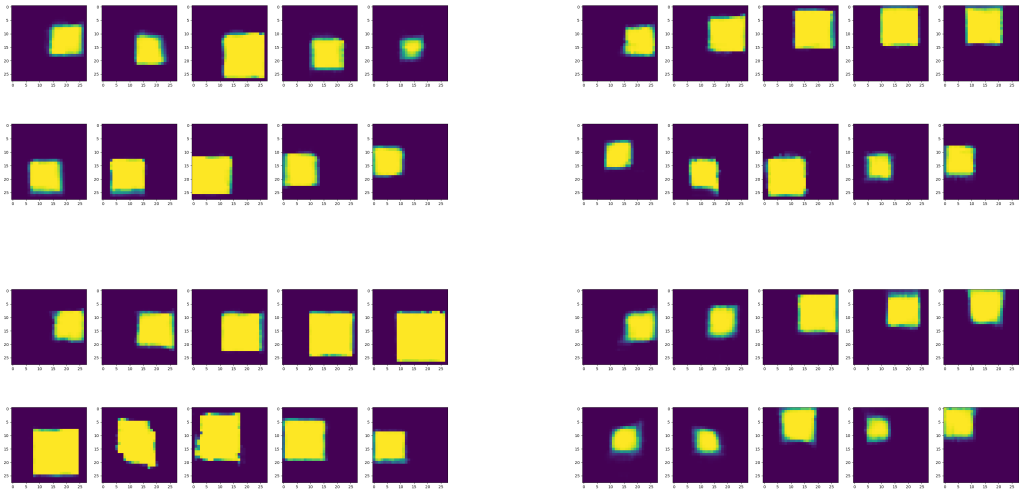


Figure C.1: Interpolating position. *Top-left*: standard normal prior, *top-right* VampPrior, *bottom-left*: SI prior, *bottom-right*: Gaussian tube.

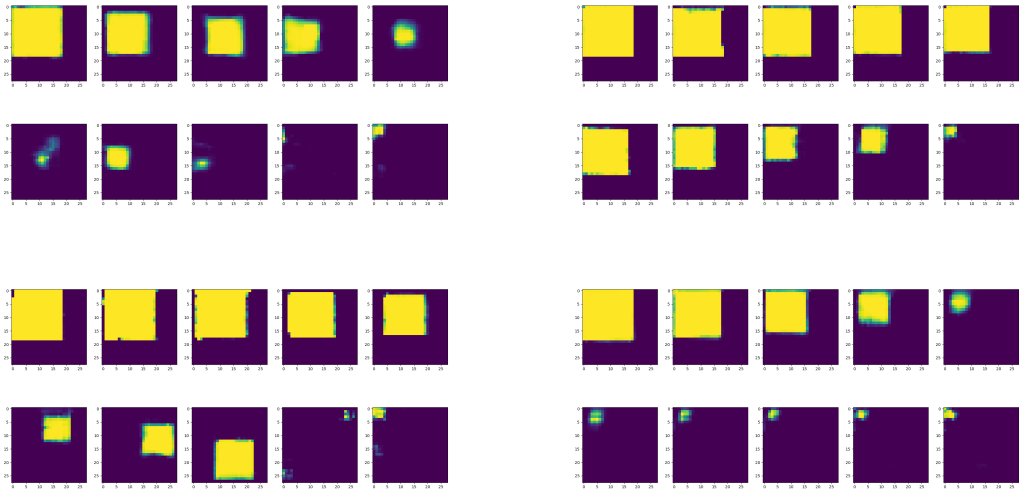


Figure C.2: Interpolating size. *Top-left*: standard normal prior, *top-right* VampPrior, *bottom-left*: SI prior, *bottom-right*: Gaussian tube.

latent space may not have the same interpolation properties as the original model.

So, the interpolation heuristic presupposes a particular smoothness to the generative model,

but the fact that we can often successfully interpolate across the latent space in trained models is interesting and may say something about how neural networks organize themselves as they train.

Appendix D

Flow-based prior

Here we propose the use of a flow-based prior distribution in variational autoencoders and argue that in contrast to using a simple prior and flexible posterior approximation, it is better to use a flexible prior and a simple variational approximation.

As we have seen, the ELBO can be written as

$$\begin{aligned}\mathcal{L}(x) &= \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)] \\ &= \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}(q(z|x)||p(z)),\end{aligned} \tag{D.1}$$

where we have shown before that the gap between the variational lower bound and the marginal log-likelihood represents the difference between the variational distribution and the true posterior

$$\log p(x) = \mathrm{KL}(q(z|x)||p(z|x)) + \mathcal{L}(x). \tag{D.2}$$

This serves as motivation for a more flexible variational distribution to enable $q(z|x)$ to come from a wider model class so it can match $p(z|x)$ more accurately and make the ELBO a better representation of the marginal log-likelihood. However, we argue two points: 1. in the interest of an interpretable latent space, a simple variational distribution is desirable 2. the VAE's complexity penalties may be easier to minimize when $p(z)$ is more complicated than $q(z|x)$.

The first point is straightforward. For the second point, let us consider an alternate expression for the average ELBO over a dataset given by [Hoffman and Johnson, 2016]:

$$\mathcal{L}(\theta, \phi) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q(z|x_n)}[\log p(x_n|z)] - (\log N - \mathbb{E}_{q(z)}[\mathbb{H}[q(n|z)]]) - \mathrm{KL}(q(z)||p(z)) \tag{D.3}$$

where $\mathbb{H}[\cdot]$ is entropy, $q(n|x) = \frac{q(z|x_n)p_{\mathrm{data}}(x_n)}{q(z)} = \frac{q(z|x_n)}{\sum_{i=1}^{N} q(z|x_i)}$, and $\mathrm{MI}[\cdot, \cdot]$ is mutual information.

[Hoffman and Johnson, 2016] notes that the mutual information term is consistently close to its maximum of $\log N$, indicating a low amount of overlap between $q(z|x)$. If $q(z|x)$ is multimodal and $p(z)$ is simple, to create a good overlap between $q(z)$ and $p(z)$, the different modes of $q(z|x)$ may have to intersperse themselves to match the smooth density of $p(z)$, but this would likely decrease the information in the latent code, unless the encoder model and/or flow model were extremely precise and the increase in proximity of modes did not lead to higher density overlap and thus loss of information. Since it has been shown that the latent space organizes itself to maximize mutual information, the latent space may 'refuse' to overlap the modes of $q(z)$, thus keeping the KL penalty high.

When $q(z|x)$ is simple and $p(z)$ is multimodal, the picture is far less complicated. Each $q(z|x)$ can correspond a mode in $p(z)$, giving rise to a high degree of overlap between marginal distributions but still a high degree of separation between latent encodings.

D.1 Flow-based prior

Flow-based models [Rezende and Mohamed, 2015; Dinh et al., 2017; Kingma and Dhariwal, 2018] make $q(z|x)$ more flexible by passing $z_0 \sim q_0(z|x)$ through a number of invertible functions:

$$z = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0), \tag{D.4}$$

giving

$$\log q(z|x) = \log q_0(z_0) - \sum_{i=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z_k} \right|, \tag{D.5}$$

where $z_k \triangleq f_k \circ f_{k-1} \circ \cdots f_1(z_0)$.

An important aspect of this framework is the Law of the Unconscious Statistician (LOTUS), given by

$$\mathbb{E}_{q(z)}[g(z)] = \mathbb{E}_{q_0(z_0)}[g(f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0))], \tag{D.6}$$

which enables us to take expectations with respect to $q(z)$ without explicitly knowing $q(z)$. This makes the ELBO easy to calculate:

$$\begin{aligned}\mathcal{L}(x) &= \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)] \\ &= \mathbb{E}_{q_0(z_0)}\left[\log p(x,z) - \log q_0(z_0) + \sum_{i=1}^{K} \log \left|\det \frac{\partial f_k}{\partial z_k}\right|\right].\end{aligned} \tag{D.7}$$

If the prior instead of the variational distribution is augmented with flows, we cannot use LOTUS as given, since our expectation is over $q(z|x)$ rather than $p(z)$. However, given $z \sim q(z|x)$, we can invert all of our flows to find the corresponding z_0 that gives $z = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)$:

$$z_0 = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_K^{-1}(z). \tag{D.8}$$

Now we can use LOTUS for $p(z)$:

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)}\left[\log p_0(z_0) - \sum_{i=1}^{K} \log \left|\det \frac{\partial f_k}{\partial z_k}\right| + \log p(x|z) - \log q(z|x)\right] \tag{D.9}$$

$$= \mathbb{E}_{q(z|x)}\left[\log p_0(z_0) + \sum_{i=1}^{K} \log \left|\det \frac{\partial f_k^{-1}}{\partial z_k}\right| + \log p(x|z) - \log q(z|x)\right]. \tag{D.10}$$

We use as our flow functions the affine transformations used in Real-NVP [Dinh et al., 2017], as their inverses have a simple closed form and are computationally easy to evaluate. Note that in Eqn (D.10), only the inverse needs to be calculated, so it is computationally equivalent to calculating the standard forward direction.

Sadly, however, we did not find an improvement in using a flow-based $p(z)$ over using a flow-based $q(z|x)$ in our experiments.