Title
Efficient Methods for Unsupervised Learning of Probabilistic Models

Author
Sohl-Dickstein, Jascha

Publication Date
2012

Peer reviewed|Thesis/dissertation

Efficient Methods for Unsupervised Learning of Probabilistic Models

by

Jascha Sohl-Dickstein

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Biophysics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Bruno A. Olshausen, Co-Chair
Professor Michael R. DeWeese, Co-Chair
Professor Friedrich T. Sommer
Professor Stanley Klein

Spring 2012

Efficient Methods for Unsupervised Learning of Probabilistic Models

by

Jascha Sohl-Dickstein

Abstract

Efficient Methods for Unsupervised Learning of Probabilistic Models

by

Jascha Sohl-Dickstein

Doctor of Philosophy in Biophysics

University of California, Berkeley

Professor Bruno A. Olshausen, Co-Chair

Professor Michael R. DeWeese, Co-Chair

High dimensional probabilistic models are used for many modern scientific and engineering data analysis tasks. Interpreting neural spike trains, compressing video, identifying features in DNA microarrays, and recognizing particles in high energy physics all rely upon the ability to find and model complex structure in a high dimensional space. Despite their great promise, high dimensional probabilistic models are frequently computationally intractable to work with in practice. In this thesis I develop solutions to overcome this intractability, primarily in the context of energy based models.

A common cause of intractability is that model distributions cannot be analytically normalized. Probabilities can only be computed up to a constant, making training exceedingly difficult. To solve this problem I propose 'minimum probability flow learning', a variational technique for parameter estimation in such models. The utility of this training technique is demonstrated in the case of an Ising model, a Hopfield auto-associative memory, an independent component analysis model of natural images, and a deep belief network.

A second common difficulty in training probabilistic models arises when the parameter space is ill-conditioned. This makes gradient descent optimization slow and impractical, but can be alleviated using the natural gradient. I show here that the natural gradient can be related to signal whitening, and provide specific prescriptions for applying it to learning problems.

It is also difficult to evaluate the performance of models that cannot be analytically normalized, providing a particular challenge to hypothesis testing and model comparison. To overcome this, I introduce a method termed 'Hamiltonian annealed importance sampling,' which more efficiently estimates the normalization constant of non-analytically-normalizable models. This method is then used to calculate and compare the log likelihoods of several state of the art probabilistic models of natural image patches.

Finally, many tasks performed with a trained probabilistic model (for instance, image denoising or inpainting and speech recognition) involve generating samples from the model distribution, which is typically a very computationally expensive process. I introduce a modification to Hamiltonian Monte Carlo sampling that reduces the tendency of sampling trajectories to double back on themselves, and enables statistically independent samples to be generated more rapidly.

Taken together, it is my hope that these contributions will help scientists and engineers to build and manipulate probabilistic models.

Acknowledgements

Thank you to my advisor Bruno Olshausen, for countless thoughtful and inspiring conversations, and for giving me the freedom to pursue my interests; my mentor Mike DeWeese, for long nights working and innumerable helpful conversations; Tony Bell for identifying the interesting questions; Fritz Sommer for many interesting conversations, and a supply of reading material; Jack Culpepper, Peter Battaglino, Charles Cadieu, Jimmy Wang, Chris Hillar, Kilian Koepsell, Amir Khosrowshahi, Urs Koester, Pierre Garrigues, and the rest of the Redwood Center for diverse and esoteric interests, many, many fascinating conversations, and useful feedback.

Contents

Chapter 1

Introduction

Scientists and engineers increasingly confront large and complex data sets that defy traditional modeling and analysis techniques. For example, fitting well-established probabilistic models from physics to population neural activity recorded in retina [Schneidman *et al.*, 2006; Shlens *et al.*, 2006; Schneidman *et al.*, 2006] or cortex [Tang *et al.*, 2008; Marre *et al.*, 2009; Yu *et al.*, 2008] is currently impractical for populations of more than about 100 neurons [Broderick *et al.*, 2007]. Similar difficulties occur in many other fields, including computer science [MacKay, 2002], genomics [Chou and Voit, 2009], and physics [Aster *et al.*, 2005]. Thus, development of new techniques to train, evaluate, and sample from complex probabilistic models is of fundamental importance to many scientific and engineering disciplines.

This thesis identifies and addresses a number of important difficulties that are encountered when working with these models. I focus on energy-based models which cannot be normalized in closed form, posing unique challenges for learning. I begin this chapter with a review of probabilistic models, current state of the art parameter estimation methods, and techniques for estimating intractable normalization constants. I end this chapter in Section 1.4 with a summary of the contributions made in this thesis, all of which improve our ability to evaluate, train, or work with challenging probabilistic models.

1.1 Parameter Estimation in Probabilistic Models

The common goal of parameter estimation is to find the parameters that cause a probabilistic model to best agree with a list $\mathcal{D}$ of (assumed iid) observations of the state of a system. In this section we provide formalism for writing data and model distributions, introduce the canonical Kullback-Leibler (KL) divergence objective for parameter estimation, and present a number of relevant parameter estimation techniques.

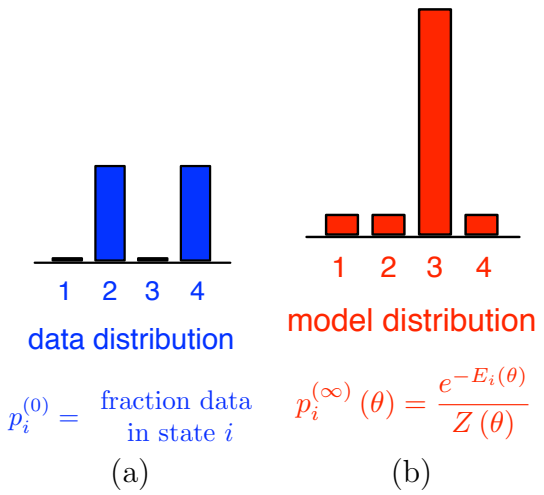


Figure 1.1: Both data and model distributions can be viewed as vectors, with a dimensionality equal to the number of possible states, and a value for each entry equal to the probability of the corresponding state. **(a)** illustrates a data distribution. In this case there were an equal number of observations of states 2 and 4, and no observations of states 1 and 3, so states 2 and 4 each have probability 0.5. **(b)** illustrates a model distribution parameterized by θ, with the probability assigned to each state i determined by the energy $E_i(\theta)$ assigned to that state and by a normalization constant $Z(\theta)$.

1.1.1 Distributions

1.1.1.1 Discrete Distributions

The data distribution is represented by a vector $\mathbf{p}^{(0)}$, with $p_i^{(0)}$ the fraction of the observations $\mathcal{D}$ in state i. The superscript (0) represents time $t = 0$ under system dynamics (which will be described for MPF in Section 2.1). For example, in a two variable binary system, $\mathbf{p}^{(0)}$ would have four entries representing the fraction of the data in states 00, 01, 10 and 11 (Figure 1.1).

Our goal is to find the parameters θ that cause a model distribution $\mathbf{p}^{(\infty)}(\theta)$ to best match the data distribution $\mathbf{p}^{(0)}$. The superscript (∞) on the model distribution indicates that this is the equilibrium distribution reached after running the dynamics (again described for MPF in Section 2.1) for infinite time. Without loss of generality, we assume the model distribution is of the form

$$p_i^{(\infty)}(\theta) = \frac{\exp\left(-E_i(\theta)\right)}{Z(\theta)}, \tag{1.1}$$

where $\mathbf{E}(\theta)$ is referred to as the energy function, and the normalizing factor $Z(\theta)$ is the

partition function,

$$Z(\theta) = \sum_i \exp\left(-E_i(\theta)\right) \tag{1.2}$$

(this can be thought of as a Boltzmann distribution of a physical system with $k_B T$ set to 1).

1.1.1.2 Continuous Distributions

Data and model distributions over a continuous state space $\mathbf{x} \in \mathcal{R}^d$ take the forms,

$$p^{(0)}(\mathbf{x}) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}' \in \mathcal{D}} \delta(\mathbf{x} - \mathbf{x}') \tag{1.3}$$

$$p^{(\infty)}(\mathbf{x}; \theta) = \frac{\exp\left(-E(\mathbf{x}; \theta)\right)}{Z(\theta)}, \tag{1.4}$$

respectively, where $|\mathcal{D}|$ is the number of observations, $\delta(\cdot)$ is the Dirac delta function, and $Z(\theta)$ is the partition function

$$Z(\theta) = \int d\mathbf{x} \exp\left(-E(\mathbf{x}; \theta)\right). \tag{1.5}$$

1.1.2 Kullback-Leibler (KL) Divergence

The standard goodness of fit measure of a model distribution to a data distribution is the KL divergence between data and model distributions [Cover *et al.*, 1991],

$$D_{KL}\left(\mathbf{p}^{(0)} \middle\| \mathbf{p}^{(\infty)}(\theta)\right) = \sum_i p_i^{(0)} \log p_i^{(0)} - \sum_i p_i^{(0)} \log p_i^{(\infty)}(\theta). \tag{1.6}$$

Because the first term in 1.6 is constant, and the second term is the negative log likelihood of the model distribution, finding the parameters which minimize the KL divergence $D_{KL}\left(\mathbf{p}^{(0)} \middle\| \mathbf{p}^{(\infty)}(\theta)\right)$ is equivalent to finding the parameters which minimize the negative log likelihood, and which maximize the likelihood. Given a list of data points $\mathcal{D}$,

$$\operatorname{argmin} D_{KL}\left(\mathbf{p}^{(0)} \middle\| \mathbf{p}^{(\infty)}(\theta)\right) = \operatorname{argmin}\left[-\sum_i p_i^{(0)} \log p_i^{(\infty)}(\theta)\right] \tag{1.7}$$

$$= \operatorname{argmin}\left[-\log L(\theta)\right] = \operatorname{argmax} L(\theta) \tag{1.8}$$

$$L(\theta) = \prod_{i \in \mathcal{D}} p_i^{(\infty)}(\theta). \tag{1.9}$$

The gradient of the log likelihood is

$$\frac{\partial \left[\log L\left(\theta\right)\right]}{\partial \theta} = -\sum_i p_i^{(0)} \frac{\partial E_i\left(\theta\right)}{\partial \theta} + \sum_i p_i^{(\infty)}\left(\theta\right) \frac{\partial E_i\left(\theta\right)}{\partial \theta}. \tag{1.10}$$

1.1.3 Parameter Estimation Techniques

Exact parameter estimation involves evaluation of $D_{KL}\left(\mathbf{p}^{(0)} \middle\| \mathbf{p}^{(\infty)}\left(\theta\right)\right)$ or its derivatives. Unfortunately, this involves evaluating $Z\left(\theta\right)$, which includes the sum over all system states in Equation 1.5, or a similar integral in the case of a continuous state space. This sum is intractable for most systems of a reasonable size - for instance involving 2^{100} terms for a 100 bit binary system. For this reason, exact parameter estimation is frequently impractical.

Many approaches exist for approximate parameter estimation, including mean field theory and its expansions, variational Bayes techniques and a variety of sampling or numerical integration based methods [Tanaka, 1998; Kappen and Rodriguez, 1997; Jaakkola and Jordan, 1997; Haykin, 2008]. The approaches which most closely relate to the new techniques introduced in this thesis include contrastive divergence (CD), developed by Hinton, Welling and Carreira-Perpiñán [Welling and Hinton, 2002; Hinton, 2002; Carreira-Perpiñán and Hinton, 2004], Hyvärinen's score matching (SM) [Hyvärinen, 2005], Besag's pseudo-likelihood (PL) [Besag, 1975], Lyu's Minimum KL Contraction [Lyu, 2011], and the minimum velocity learning framework proposed by Movellan [Movellan, 2008a; Movellan, 2008b; Movellan and McClelland, 1993].

1.1.3.1 Contrastive Divergence

Contrastive divergence [Welling and Hinton, 2002; Hinton, 2002] is a variation on steepest gradient descent of the maximum (log) likelihood (ML) objective function. Rather than integrating over the full model distribution, CD approximates the partition function term in the gradient by averaging over the distribution obtained after taking a few, or only one, Markov chain Monte Carlo (MCMC) steps away from the data distribution (Equation 1.11). CD is frequently abbreviated CD-k, where k is the number of MCMC steps taken away from the data distribution. Qualitatively, one can imagine that the data distribution is contrasted against a distribution that has evolved only a small distance towards the model distribution, whereas it would be contrasted against the true model distribution in traditional MCMC approaches. Although CD is not guaranteed to converge to the right answer, or even to a fixed point, it has proven to be an effective and fast heuristic for parameter estimation [MacKay, 2001; Yuille, 2005]. The CD-k update rule can be written

$$\Delta\theta_{CD} \propto -\sum_i p_i^{(0)} \frac{\partial E_i\left(\theta\right)}{\partial \theta} + \sum_i p_i^{(k)} \frac{\partial E_i\left(\theta\right)}{\partial \theta}, \tag{1.11}$$

where $p^{(k)}$ is the distribution resulting after applying k MCMC updates to samples from $p^{(0)}$. This update rule should be compared against the gradient of the log likelihood in Equation 1.10.

1.1.3.2 Score Matching

Score matching [Hyvärinen, 2005] is a method that learns parameters in a probabilistic model with a continuous state space using only derivatives of the energy function evaluated over the data distribution. This sidesteps the need to explicitly sample or integrate over the model distribution. In score matching one minimizes the expected square distance of the score function with respect to spatial coordinates given by the data distribution from the similar score function given by the model distribution. The score function is the gradient of the log likelihood. A number of connections have been made between score matching and other learning techniques [Hyvärinen, 2007a; Sohl-Dickstein and Olshausen, 2009; Movellan, 2008a; Lyu, 2009]. The score matching objective function can be written

$$K_{SM}(\theta) = \sum_{\mathbf{x}\in\mathcal{D}} \left[\frac{1}{2} \nabla E(\mathbf{x};\theta) \cdot \nabla E(\mathbf{x};\theta) - \nabla^2 E(\mathbf{x};\theta) \right]. \tag{1.12}$$

Parameter estimation is performed by finding $\operatorname{argmin}_\theta K_{SM}(\theta)$. Performing gradient descent on $K_{SM}(\theta)$ involves computing 3rd derivatives of $E(\mathbf{x};\theta)$, which is frequently unwieldy.

1.1.3.3 Pseudolikelihood

Pseudolikelihood [Besag, 1975] approximates the joint probability distribution of a collection of random variables by a computationally tractable product of conditional distributions, where each factor is the distribution of a single random variable conditioned on the others. This approach often leads to surprisingly good parameter estimates, despite the extreme nature of the approximation. Recent work suggests that pseudolikelihood is a consistent estimator of model parameters [Lyu, 2011], meaning that if the data distribution has the same form as the model distribution then in the limit of infinite data the exact correct distribution will be recovered. The pseudolikelihood objective function can be written

$$K_{PL}(\theta) = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{m} \log p\left(x_m | \mathbf{x}_{\backslash m}; \theta\right), \tag{1.13}$$

where m indexes the dimensions of the state space, and the expression $p\left(x_m|\mathbf{x}_{\backslash m}\right)$ indicates the probability distribution over the mth dimension of the state space conditioned on the remaining dimensions. For clarity we have written the pseudolikelihood objective function for a continuous state space, but it is defined for both continuous and discrete state spaces.

1.1.3.4 Minimum Velocity Learning

Minimum velocity learning is an approach recently proposed by Movellan [Movellan, 2008a] that recasts a number of the ideas behind CD, treating the minimization of the initial dynamics away from the data distribution as the goal itself rather than a surrogate for it. Rather than directly minimize the difference between the data and the model, Movellan's proposal is to introduce system dynamics that have the model as their equilibrium distribution, and minimize the initial flow of probability away from the data under those dynamics. If the model looks exactly like the data there will be no flow of probability, and if model and data are similar the flow of probability will tend to be minimal. Movellan applies this intuition to the specific case of distributions over continuous state spaces evolving via diffusion dynamics, and recovers the score matching objective function (Equation 1.12). The velocity in minimum velocity learning is the difference in average drift velocities between particles diffusing under the model distribution and particles diffusing under the data distribution.

1.1.3.5 Minimum KL contraction

Minimum KL contraction [Lyu, 2011] involves applying a special class of mapping (a contraction mapping) to both the data and model distributions, and minimizing the amount by which this mapping shrinks the KL divergence between the data and model distributions. As the KL divergence between the data and model distributions becomes more similar there is less room for the contraction mapping to further shrink it, and the KL contraction objective becomes smaller. Like minimum probability flow (introduced in Chapter 2), minimum KL contraction appears to be a generalization of a number of existing parameter estimation techniques based on "local" information about the model distribution.

1.2 Hamiltonian Monte Carlo Sampling

Generating samples from probability distributions over high dimensional state spaces is frequently extremely expensive. Hamiltonian Monte Carlo (HMC) [Horowitz, 1991; Neal, 2010] is a family of techniques for fast sampling in continuous state spaces, which work by extending the state space to include auxiliary momentum variables, and then simulating Hamiltonian dynamics from physics in order to traverse long iso-probability trajectories which rapidly explore the state space.

In HMC, the state space $\mathbf{x} \in \mathbb{R}^M$ is expanded to include auxiliary momentum variables $\mathbf{v} \in \mathbb{R}^M$ with a simple independent distribution,

$$p(\mathbf{v}) = \frac{\exp\left[-\frac{1}{2}\mathbf{v}^T\mathbf{v}\right]}{(2\pi)^{\frac{M}{2}}}. \tag{1.14}$$

The joint distribution over $\mathbf{x}$ and $\mathbf{v}$ is then

$$p(\mathbf{x}, \mathbf{v}) = p(\mathbf{x})\, p(\mathbf{v}) = \frac{\exp\left[-H(\mathbf{x}, \mathbf{v})\right]}{Z_H} \tag{1.15}$$

$$H(\mathbf{x}, \mathbf{v}) = E(\mathbf{x}) + \frac{1}{2}\mathbf{v}^T\mathbf{v}, \tag{1.16}$$

where $H(\mathbf{x}, \mathbf{v})$ is the total energy, Z_H is a normalization constant, and $E(\mathbf{x})$ and $\frac{1}{2}\mathbf{v}^T\mathbf{v}$ are analogous to the potential and kinetic energies in a physical system.

Sampling alternates between drawing the momentum $\mathbf{v}$ from its marginal distribution $p(\mathbf{v})$, and simulating Hamiltonian dynamics for the joint system described by $H(\mathbf{x}, \mathbf{v})$. Hamiltonian dynamics are described by the differential equations

$$\dot{\mathbf{x}} = \frac{\partial H(\mathbf{x}, \mathbf{v})}{\partial \mathbf{v}} = \mathbf{v} \tag{1.17}$$

$$\dot{\mathbf{v}} = -\frac{\partial H(\mathbf{x}, \mathbf{v})}{\partial \mathbf{x}} = -\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}. \tag{1.18}$$

Because Hamiltonian dynamics conserve the total energy $H(\mathbf{x}, \mathbf{v})$ and thus the joint probability $p(\mathbf{x}, \mathbf{v})$, and preserve volume in the joint space of $\mathbf{x}$ and $\mathbf{v}$, new samples proposed by integrating Equations 1.17 and 1.18 can be accepted with probability one, yet also have traversed a large distance from the previous sample. HMC thus allows independent samples to be rapidly drawn from $p(\mathbf{x}, \mathbf{v})$. Because $p(\mathbf{x}, \mathbf{v})$ is factorial, samples from $p(\mathbf{x})$ can be recovered by discarding the $\mathbf{v}$ variables and taking the marginal distribution over $\mathbf{x}$. Additional issues which must be addressed in implementation involve choosing a numerical integration scheme for the dynamics, and correctly accounting for discretization errors.

1.3 Annealed Importance Sampling

Annealed Importance Sampling (AIS) [Neal, 2001] is a sequential Monte Carlo method [Moral *et al.*, 2006] which allows the partition function of a non-analytically-normalizable distribution to be estimated in an unbiased fashion. This is accomplished by starting at a distribution with a known normalization, and gradually transforming it into the distribution of interest through a chain of Markov transitions. Its practicality depends heavily on the chosen Markov transitions. Here we review the derivations of both importance sampling and annealed importance sampling. An extension of annealed importance sampling to better incorporate Hamiltonian Monte Carlo is presented in Chapter 5.

1.3.1 Importance Sampling

Importance sampling [Kahn and Marshall, 1953] allows an unbiased estimate $\hat{Z}_p$ of the partition function (or normalization constant) Z_p of a non-analytically-normalizable target distribution $p(\mathbf{x})$ over $\mathbf{x} \in \mathbb{R}^M$,

$$p(\mathbf{x}) = \frac{e^{-E_p(\mathbf{x})}}{Z_p} \tag{1.19}$$

$$Z_p = \int d\mathbf{x}\; e^{-E_p(\mathbf{x})}, \tag{1.20}$$

to be calculated. This is accomplished by averaging over samples $\mathcal{S}_q$ from a proposal distribution $q(\mathbf{x})$,

$$q(\mathbf{x}) = \frac{e^{-E_q(\mathbf{x})}}{Z_q} \tag{1.21}$$

$$Z_p = \int d\mathbf{x}\; q(\mathbf{x}) \frac{e^{-E_p(\mathbf{x})}}{q(\mathbf{x})} \tag{1.22}$$

$$\hat{Z}_p = \frac{1}{|\mathcal{S}_q|} \sum_{x \in \mathcal{S}_q} \frac{e^{-E_p(\mathbf{x})}}{q(\mathbf{x})}, \tag{1.23}$$

where $|\mathcal{S}_q|$ is the number of samples. $q(\mathbf{x})$ is chosen to be easy both to sample from and to evaluate exactly, and must have support everywhere that $p(\mathbf{x})$ does. Unfortunately, unless $q(\mathbf{x})$ has significant mass everywhere $p(\mathbf{x})$ does, it takes an impractically large number of samples from $q(\mathbf{x})$ for $\hat{Z}_p$ to accurately approximate Z_p[1].

1.3.2 Annealed Importance Sampling

Annealed importance sampling [Neal, 2001] extends the state space $\mathbf{x}$ to a series of vectors, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N\}$, $\mathbf{x}_n \in \mathbb{R}^M$. It then transforms the proposal distribution $q(\mathbf{x})$ to a distribution $Q(\mathbf{X})$ over $\mathbf{X}$, by setting $q(\mathbf{x})$ as the distribution over $\mathbf{x}_1$ and then multiplying by a series of Markov transition distributions,

$$Q(\mathbf{X}) = q(\mathbf{x}_1) \prod_{n=1}^{N-1} T_n(\mathbf{x}_{n+1} | \mathbf{x}_n), \tag{1.24}$$

[1] The expected variance of the estimate $\hat{Z}_p$ is given by an α-divergence between $p(\mathbf{x})$ and $q(\mathbf{x})$, times a constant and plus an offset - see [Minka, 2005].

where $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$ represents a *forward* transition distribution from $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$. The target distribution $p(\mathbf{x})$ is similarly transformed to become a reverse chain $P(\mathbf{X})$, starting at $\mathbf{x}_N$, over $\mathbf{X}$,

$$P(\mathbf{X}) = \frac{e^{-E_p(\mathbf{x}_N)}}{Z_p} \prod_{n=1}^{N-1} \widetilde{T}_n(\mathbf{x}_n|\mathbf{x}_{n+1}), \tag{1.25}$$

where $\widetilde{T}_n(\mathbf{x}_n|\mathbf{x}_{n+1})$ is a *reverse* transition distribution from $\mathbf{x}_{n+1}$ to $\mathbf{x}_n$. The transition distributions are, by definition, normalized (eg, $\int d\mathbf{x}_{n+1} T_n(\mathbf{x}_{n+1}|\mathbf{x}_n) = 1$).

In a similar fashion to Equations 1.22 and 1.23, samples $\mathcal{S}_Q$ from the forward proposal chain $Q(\mathbf{X})$ can be used to estimate the partition function Z_p,

$$Z_p = \int d\mathbf{x}_N \; e^{-E_p(\mathbf{x}_N)} \tag{1.26}$$

$$= \int d\mathbf{x}_N \; e^{-E_p(\mathbf{x}_N)} \int d\mathbf{x}_{N-1} \; \widetilde{T}_{N-1}(\mathbf{x}_{N-1}|\mathbf{x}_N) \cdots \int d\mathbf{x}_1 \; \widetilde{T}_1(\mathbf{x}_1|\mathbf{x}_2) \tag{1.27}$$

(note that all integrals but the first in Equation 1.27 go to 1)

$$Z_p = \int d\mathbf{X} \; Q(\mathbf{X}) \frac{e^{-E_p(\mathbf{x}_N)}}{Q(\mathbf{X})} \widetilde{T}_{N-1}(\mathbf{x}_{N-1}|\mathbf{x}_N) \cdots \; \widetilde{T}_1(\mathbf{x}_1|\mathbf{x}_2) \tag{1.28}$$

$$\hat{Z}_p = \frac{1}{|\mathcal{S}_Q|} \sum_{X \in \mathcal{S}_Q} \frac{e^{-E_p(\mathbf{x}_N)}}{q(\mathbf{x}_1)} \frac{\widetilde{T}_1(\mathbf{x}_1|\mathbf{x}_2)}{T_1(\mathbf{x}_2|\mathbf{x}_1)} \cdots \frac{\widetilde{T}_{N-1}(\mathbf{x}_{N-1}|\mathbf{x}_N)}{T_{N-1}(\mathbf{x}_N|\mathbf{x}_{N-1})}. \tag{1.29}$$

In order to further define the transition distributions, Neal introduces intermediate distributions $\pi_n(\mathbf{x})$ between $q(\mathbf{x})$ and $p(\mathbf{x})$,

$$\pi_n(\mathbf{x}) = \frac{e^{-E_{\pi_n}(x)}}{Z_{\pi_n}} \tag{1.30}$$

$$E_{\pi_n}(\mathbf{x}) = (1-\beta_n) E_q(\mathbf{x}) + \beta_n E_p(\mathbf{x}), \tag{1.31}$$

where the mixing fraction $\beta_n = \frac{n}{N}$ for all the results in this thesis. $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$ is then chosen to be any Markov chain transition for $\pi_n(\mathbf{x})$, meaning that it leaves $\pi_n(\mathbf{x})$ invariant

$$T_n \circ \pi_n = \pi_n. \tag{1.32}$$

The reverse direction transition distribution $\widetilde{T}_n(\mathbf{x}_n|\mathbf{x}_{n+1})$ is set to the reversal of $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$,

$$\widetilde{T}_n(\mathbf{x}_n|\mathbf{x}_{n+1}) = T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)\frac{\pi_n(\mathbf{x}_n)}{\pi_n(\mathbf{x}_{n+1})}. \tag{1.33}$$

Equation 1.29 thus reduces to

$$\hat{Z}_p = \frac{1}{|\mathcal{S}_Q|}\sum_{X\in\mathcal{S}_Q}\frac{e^{-E_p(\mathbf{x}_N)}}{q(\mathbf{x}_1)}\frac{\pi_1(\mathbf{x}_1)}{\pi_1(\mathbf{x}_2)}\cdots\frac{\pi_{N-1}(\mathbf{x}_{N-1})}{\pi_{N-1}(\mathbf{x}_N)} \tag{1.34}$$

$$= \frac{1}{|\mathcal{S}_Q|}\sum_{X\in\mathcal{S}_Q}\frac{e^{-E_p(\mathbf{x}_N)}}{q(\mathbf{x}_1)}\frac{e^{-E_{\pi_1}(x_1)}}{e^{-E_{\pi_1}(x_2)}}\cdots\frac{e^{-E_{\pi_{N-1}}(x_{N-1})}}{e^{-E_{\pi_{N-1}}(x_N)}}. \tag{1.35}$$

If the number of intermediate distributions N is large, and the transition distributions $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$ and $\widetilde{T}_n(\mathbf{x}_n|\mathbf{x}_{n+1})$ mix effectively, then the distributions over intermediate states $\mathbf{x}_n$ will be nearly identical to $\pi_n(\mathbf{x}_n)$ in both the forward and backward chains. $P(\mathbf{X})$ and $Q(\mathbf{X})$ will then be extremely similar to one another, and the variance in the estimate $\hat{Z}_p$ will be extremely low[2]. If the transitions $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$ do a poor job mixing, then the marginal distributions over $\mathbf{x}_n$ under $P(\mathbf{X})$ and $Q(\mathbf{X})$ will look different from $\pi_n(\mathbf{x}_n)$. The estimate $\hat{Z}_p$ will still be unbiased, but with a potentially larger variance. Thus, to make AIS practical, it is important to choose Markov transitions $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$ for the intermediate distributions $\pi_n(\mathbf{x})$ that mix quickly.

1.4 Contributions of this Thesis

In this thesis I attempt to solve several of the problems that arise in probabilistic modeling. I began in Chapter 1 by reviewing existing techniques for working with intractable probabilistic models.

One of the most significant problems working with probabilistic models is that the majority of them cannot be analytically normalized. Therefore the probabilities they assign to states cannot be exactly computed, and are expensive even to approximate. Training a model with an intractable normalization constant is extremely difficult using traditional methods based on sampling. I present an alternative technique for parameter estimation in such models, Minimum Probability Flow (MPF), in Chapter 2.

In Chapter 3 I present experiments demonstrating the effectiveness of MPF for a number of applications. These quantitative results include comparisons of estimation speed and

[2] There is a direct mapping between annealed importance sampling and the Jarzynski equality in non-equilibrium thermodynamics — see [Jarzynski, 1997]. It follows from this mapping, and the reversibility of quasistatic processes, that the variance in $\hat{Z}_p$ can be made to go to 0 if the transition from $q(\mathbf{x}_1)$ to $p(\mathbf{x}_N)$ is sufficiently gradual.

quality for an Ising model, the application of MPF to storing memories in a Hopfield auto-associative memory, and an evaluation of estimation quality for an Independent Component Analysis (ICA) model and a Deep Belief Network (DBN).

Difficulties in training probabilistic models can stem from ill conditioning of the model's parameter space as well as from an inability to analytically normalize the model. In Chapter 4 I review how an ill conditioned parameter space can undermine learning, and present a novel interpretation of the natural gradient, a common technique for dealing with this ill conditioning. In addition, I present tricks and specific prescriptions for applying the natural gradient to learning problems.

Even after a probabilistic model has been trained, it remains difficult to objectively judge and compare its performance to that of other models unless it can be normalized. To address this, in Chapter 5 Hamiltonian Annealed Importance Sampling (HAIS) is presented. This is a method which can be used for more efficient log likelihood estimation which combines Hamiltonian Monte Carlo (HMC) with Annealed Importance Sampling (AIS). It is then applied to compare the log likelihoods of several state of the art probabilistic models of natural image patches.

Finally, many of the tasks commonly performed with probabilistic models, for instance image denoising or inpainting [Roth and Black, 2005] and speech recognition [Zweig, 1998], require samples from the model distribution. Generating those samples has a high computational cost, often making it the bottleneck in a machine learning task. In Chapter 6 an extension to HMC sampling is introduced which reduces the frequency with which sampling trajectories double back on themselves, and thus enables statistically independent samples to be generated more rapidly.

Additional research involving high dimensional probabilistic models, not incorporated into this thesis, includes developing multilinear generative models for natural scenes [Culpepper *et al.*, 2011], training Lie groups to describe the transformations which occur in natural video [Sohl-Dickstein *et al.*, 2010; Wang *et al.*, 2011], exploring the statistical structure of MRI and CT scans of breast tissue [Abbey *et al.*, 2009], applying a super-resolution algorithm to images from the Mars Exploration Rover Panoramic Camera [Hayes *et al.*, 2011; Grotzinger *et al.*, 2005; Bell *et al.*, 2004b; Bell *et al.*, 2004a], photometric modeling of Martian dust [Kinch *et al.*, 2007; Johnson *et al.*, 2006], and modeling of camera systems on the Mars Exploration Rover [Bell *et al.*, 2006; Herkenhoff *et al.*, 2003].

Chapter 2

Minimum Probability Flow

As discussed in Chapter 1, most probabilistic learning techniques require calculating the normalization factor, or partition function, of the probabilistic model in question, or at least calculating its gradient. For the overwhelming majority of models there are no known analytic solutions, and this calculation is intractable. In this chapter we will present a technique for parameter estimation in probabilistic models, even in cases where the normalization factor cannot be calculated. Material in this chapter is taken from [Sohl-Dickstein *et al.*, 2011b; Sohl-Dickstein *et al.*, 2011a; Sohl-Dickstein *et al.*, 2009].

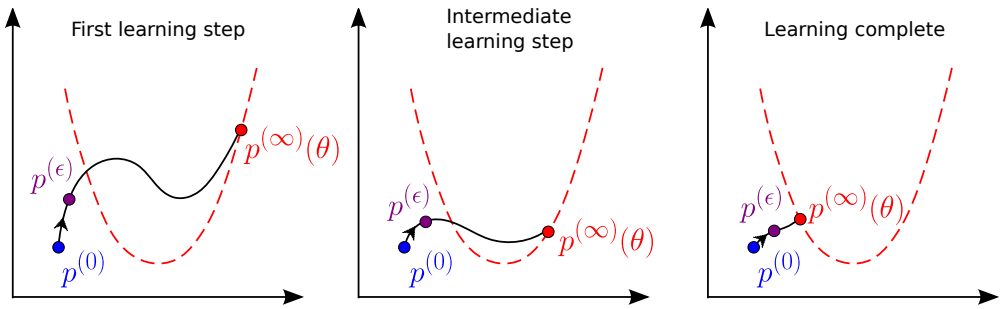
Our goal is to find the parameters that cause a probabilistic model to best agree with a list $\mathcal{D}$ of (assumed iid) observations of the state of a system. We will do this by introducing deterministic dynamics that guarantee the transformation of the data distribution into the model distribution, and then minimizing the KL divergence between the data distribution and the distribution that results from running those dynamics for a short time ϵ (see Figure 2.1). Formalism used below is introduced in Section 1.1.

2.1 Dynamics

Most Monte-Carlo algorithms rely on two core concepts from statistical physics, the first being conservation of probability as enforced by the master equation for the time evolution of a distribution $\mathbf{p}^{(t)}$ [Pathria, 1972]:

$$\dot{p}_i^{(t)} = \sum_{j \neq i} \Gamma_{ij}(\theta)\, p_j^{(t)} - \sum_{j \neq i} \Gamma_{ji}(\theta)\, p_i^{(t)}, \tag{2.1}$$

where $\dot{p}_i^{(t)}$ is the time derivative of $p_i^{(t)}$. Transition rates $\Gamma_{ij}(\theta)$, for $i \neq j$, give the rate at which probability flows from a state j into a state i. The first term of Equation (2.1) captures the flow of probability out of other states j into the state i, and the second captures flow out of i into other states j. The dependence on θ results from the requirement that the chosen



Progression of Learning

Figure 2.1: An illustration of parameter estimation using minimum probability flow (MPF). In each panel, the axes represent the space of all probability distributions. The three successive panels illustrate the sequence of parameter updates that occur during learning. The dashed red curves indicate the family of model distributions $\mathbf{p}^{(\infty)}(\theta)$ parametrized by θ. The black curves indicate deterministic dynamics that transform the data distribution $\mathbf{p}^{(0)}$ into the model distribution $\mathbf{p}^{(\infty)}(\theta)$. Under maximum likelihood learning, model parameters θ are chosen so as to minimize the Kullback–Leibler (KL) divergence between the data distribution $\mathbf{p}^{(0)}$ and the model distribution $\mathbf{p}^{(\infty)}(\theta)$. Under MPF, however, the KL divergence between $\mathbf{p}^{(0)}$ and $\mathbf{p}^{(\epsilon)}$ is minimized instead, where $\mathbf{p}^{(\epsilon)}$ is the distribution obtained by initializing the dynamics at the data distribution $\mathbf{p}^{(0)}$ and then evolving them for an infinitesimal time ϵ. Here we represent graphically how parameter updates that pull $\mathbf{p}^{(\epsilon)}$ towards $\mathbf{p}^{(0)}$ also tend to pull $\mathbf{p}^{(\infty)}(\theta)$ towards $\mathbf{p}^{(0)}$.

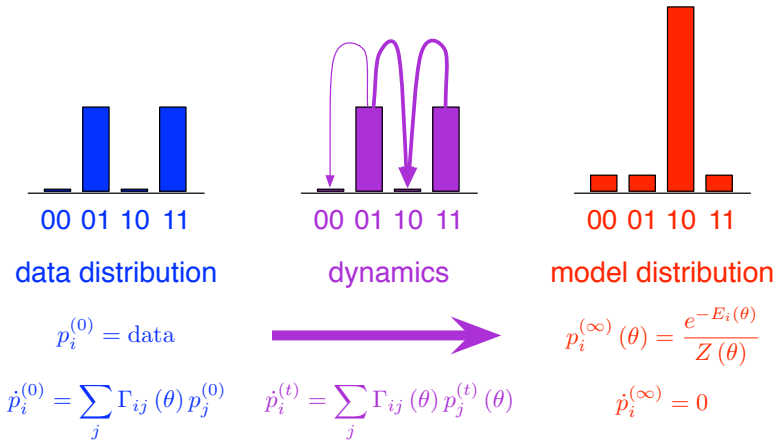


Figure 2.2: Dynamics of minimum probability flow learning. Model dynamics represented by the probability flow matrix $\mathbf{\Gamma}$ (*middle*) determine how probability flows from the empirical histogram of the sample data points (*left*) to the equilibrium distribution of the model (*right*) after a sufficiently long time. In this example there are only four possible states for the system, which consists of a pair of binary variables, and the particular model parameters favor state 10 whereas the data falls on other states.

dynamics cause $\mathbf{p}^{(t)}$ to flow to the equilibrium distribution $\mathbf{p}^{(\infty)}(\theta)$. For readability, explicit dependence on θ will be dropped except where necessary. If we choose to set the diagonal elements of $\mathbf{\Gamma}$ to obey $\Gamma_{ii} = -\sum_{j \neq i} \Gamma_{ji}$, then we can write the dynamics as

$$\dot{\mathbf{p}}^{(t)} = \mathbf{\Gamma}\mathbf{p}^{(t)} \tag{2.2}$$

(see Figure 2.2). The unique solution for $\mathbf{p}^{(t)}$ is given by[1]

$$\mathbf{p}^{(t)} = \exp(\mathbf{\Gamma}t)\,\mathbf{p}^{(0)}, \tag{2.3}$$

where $\exp(\mathbf{\Gamma}t)$ is a matrix exponential.

[1] The form chosen for $\mathbf{\Gamma}$ in Equation (2.2), coupled with the satisfaction of detailed balance and ergodicity introduced in section 2.2, guarantees that there is a unique eigenvector $\mathbf{p}^{(\infty)}$ of $\mathbf{\Gamma}$ with eigenvalue zero, and that all other eigenvalues of $\mathbf{\Gamma}$ are real and negative.

2.2 Detailed Balance

The second core concept is detailed balance,

$$\Gamma_{ji}\; p_i^{(\infty)}(\theta) = \Gamma_{ij}\; p_j^{(\infty)}(\theta)\,, \tag{2.4}$$

which states that at equilibrium the probability flow from state i into state j equals the probability flow from j into i. When satisfied, detailed balance guarantees that the distribution $\mathbf{p}^{(\infty)}(\theta)$ is a fixed point of the dynamics. Sampling in most Monte Carlo methods is performed by choosing $\mathbf{\Gamma}$ consistent with Equation 2.4 (and the added requirement of ergodicity), then stochastically running the dynamics of Equation 2.1. Note that there is no need to restrict the dynamics defined by $\mathbf{\Gamma}$ to those of any real physical process, such as diffusion.

Equation 2.4 can be written in terms of the model's energy function $\mathbf{E}(\theta)$ by substituting in Equation 1.1 for $\mathbf{p}^{(\infty)}(\theta)$:

$$\Gamma_{ji} \exp\left(-E_i(\theta)\right) = \Gamma_{ij} \exp\left(-E_j(\theta)\right). \tag{2.5}$$

$\mathbf{\Gamma}$ is underconstrained by the above equation. Introducing the additional constraint that Γ be invariant to the addition of a constant to the energy function (as the model distribution $\mathbf{p}^{(\infty)}(\theta)$ is), we choose the following form for the non-diagonal entries in $\mathbf{\Gamma}$

$$\Gamma_{ij} = g_{ij} \exp\left[\frac{1}{2}\left(E_j(\theta) - E_i(\theta)\right)\right] \qquad (i \neq j)\,, \tag{2.6}$$

where the connectivity function

$$g_{ij} = g_{ji} = \begin{cases} 0 & \text{unconnected states} \\ 1 & \text{connected states} \end{cases} \qquad (i \neq j) \tag{2.7}$$

determines which states are allowed to directly exchange probability with each other. The non-zero $\mathbf{\Gamma}$ may also be sampled from a proposal distribution rather than set via a deterministic scheme, in which case g_{ij} takes on the role of proposal distribution - see Section 2.8. g_{ij} can be set such that $\mathbf{\Gamma}$ is *extremely* sparse (see Section 2.4). Theoretically, to guarantee convergence to the model distribution, the non-zero elements of $\mathbf{\Gamma}$ must be chosen such that, given sufficient time, probability can flow between any pair of states (ergodicity).

2.3 Objective Function

Maximum likelihood parameter estimation involves maximizing the likelihood of some observations $\mathcal{D}$ under a model, or equivalently minimizing the KL divergence between the data

distribution $\mathbf{p}^{(0)}$ and model distribution $\mathbf{p}^{(\infty)}$,

$$\hat{\theta}_{\mathrm{ML}} = \underset{\theta}{\operatorname{argmin}}\, D_{KL}\left(\mathbf{p}^{(\mathbf{0})} || \mathbf{p}^{(\infty)}\left(\theta\right)\right) \tag{2.8}$$

Rather than running the dynamics for infinite time, we propose to minimize the KL divergence after running the dynamics for an infinitesimal time ϵ,

$$\hat{\theta}_{\mathrm{MPF}} = \underset{\theta}{\operatorname{argmin}}\, K\left(\theta\right) \tag{2.9}$$

$$K\left(\theta\right) = D_{KL}\left(\mathbf{p}^{(\mathbf{0})} || \mathbf{p}^{(\epsilon)}\left(\theta\right)\right). \tag{2.10}$$

For small ϵ, $D_{KL}\left(\mathbf{p}^{(\mathbf{0})} || \mathbf{p}^{(\epsilon)}\left(\theta\right)\right)$ can be approximated by a first order Taylor expansion,

$$\begin{aligned} K\left(\theta\right) \approx D_{KL}\left(\mathbf{p}^{(\mathbf{0})} || \mathbf{p}^{(\mathbf{t})}\left(\theta\right)\right)\Big|_{t=0} \\ + \epsilon \frac{\partial D_{KL}\left(\mathbf{p}^{(\mathbf{0})} || \mathbf{p}^{(\mathbf{t})}\left(\theta\right)\right)}{\partial t}\Big|_{t=0}. \end{aligned} \tag{2.11}$$

Further algebra (see Appendix A) reduces $K\left(\theta\right)$ to a measure of the flow of probability, at time $t = 0$ under the dynamics, out of data states $j \in \mathcal{D}$ into non-data states $i \notin \mathcal{D}$,

$$K\left(\theta\right) = \frac{\epsilon}{|\mathcal{D}|} \sum_{i \notin \mathcal{D}} \sum_{j \in \mathcal{D}} \Gamma_{ij} \tag{2.12}$$

$$= \frac{\epsilon}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} \sum_{i \notin \mathcal{D}} g_{ij} \exp\left[\frac{1}{2}\left(E_j\left(\theta\right) - E_i\left(\theta\right)\right)\right] \tag{2.13}$$

with gradient

$$\begin{aligned} \frac{\partial K\left(\theta\right)}{\partial \theta} = \frac{\epsilon}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} \sum_{i \notin \mathcal{D}} \left[\frac{\partial E_j\left(\theta\right)}{\partial \theta} - \frac{\partial E_i\left(\theta\right)}{\partial \theta}\right] \\ g_{ij} \exp\left[\frac{1}{2}\left(E_j\left(\theta\right) - E_i\left(\theta\right)\right)\right], \end{aligned} \tag{2.14}$$

where $|\mathcal{D}|$ is the number of observed data points. Note that Equations (2.12) and (2.14) do not depend on the partition function $Z\left(\theta\right)$ or its derivatives.

$K\left(\theta\right)$ is uniquely zero when $\mathbf{p}^{(0)}$ and $\mathbf{p}^{(\infty)}\left(\theta\right)$ are equal. This implies consistency, in that if the data comes from the model class, in the limit of infinite data $K\left(\theta\right)$ will be minimized by exactly the true θ. In addition, $K\left(\theta\right)$ is convex for all models $\mathbf{p}^{(\infty)}\left(\theta\right)$ in the exponential family - that is, models whose energy functions $\mathbf{E}\left(\theta\right)$ are linear in their parameters θ [Macke and Gerwinn, 2009] (see Appendix B). The MPF objective additionally provides an upper

bound on the log likelihood of the data if the first non-zero eigenvalue of $\mathbf{\Gamma}$ is known (see Appendix C).

2.4 Tractability

The dimensionality of the vector $\mathbf{p}^{(0)}$ is typically huge, as is that of $\mathbf{\Gamma}$ (*e.g.*, 2^d and $2^d \times 2^d$, respectively, for a d-bit binary system). Naïvely, this would seem to prohibit evaluation and minimization of the objective function. Fortunately, we need only visit those columns of Γ_{ij} corresponding to data states, $j \in \mathcal{D}$. Additionally, g_{ij} can be populated so as to connect each state j to only a small fixed number of additional states i. The cost in both memory and time to evaluate the objective function is thus $\mathcal{O}(|\mathcal{D}|)$, and does not depend on the number of system states, only on the (much smaller) number of observed data points.

2.5 Choosing the Connectivity Function g

Qualitatively, the most informative states to connect data states to are those that are most probable under the model. In discrete state spaces, nearest neighbor connectivity schemes for g_{ji} work extremely well (eg Equation 3.2 below). This is because, as learning converges, the states that are near data states become the states that are probable under the model.

2.6 Continuous State Spaces

Although we have motivated this technique using systems with a large, but finite, number of states, it generalizes to continuous state spaces. Γ_{ji}, g_{ji}, and $p_i^{(t)}$ become continuous functions $\Gamma(\mathbf{x}_j, \mathbf{x}_i)$, $g(\mathbf{x}_j, \mathbf{x}_i)$, and $p^{(t)}(\mathbf{x}_i)$. $\Gamma(\mathbf{x}_j, \mathbf{x}_i)$ can be populated stochastically and extremely sparsely, preserving the $\mathcal{O}(|\mathcal{D}|)$ cost.

In continuous state spaces, the estimated parameters are much more sensitive to the choice of $g(\mathbf{x}_j, \mathbf{x}_i)$. Practically, we have implemented MPF in continuous state spaces using the persistent particle extensions in Section 2.9, and Hamiltonian Monte Carlo (HMC) to sample the connected states.

2.7 Connection to Other Learning Techniques

2.7.1 Contrastive Divergence

The contrastive divergence update rule (introduced in Section 1.1.3.1) can be written in the form

$$\Delta\theta_{CD} \propto -\sum_{j\in\mathcal{D}}\sum_{i\notin\mathcal{D}}\left[\frac{\partial E_j(\theta)}{\partial\theta} - \frac{\partial E_i(\theta)}{\partial\theta}\right] T_{ij}, \tag{2.15}$$

where T_{ij} is the probability of transitioning from state j to state i in a single Markov chain Monte Carlo step (or k steps for CD-k). Equation 2.15 has obvious similarities to the MPF learning gradient in Equation 2.14. Thus, steepest gradient descent under MPF resembles CD updates, but with the MCMC sampling/rejection step T_{ij} replaced by a weighting factor $g_{ij}\exp\left[\frac{1}{2}\left(E_j(\theta) - E_i(\theta)\right)\right]$.

Note that this difference in form provides MPF with a well-defined objective function. One important consequence of the existence of an objective function is that MPF can readily utilize general purpose, off-the-shelf optimization packages for gradient descent, which would have to be tailored in some way to be applied to CD. This is part of what accounts for the dramatic difference in learning time between CD and MPF in some cases (see Figure 3.1).

2.7.2 Score Matching

For a continuous state space, MPF reduces to score matching (introduced in Section 1.1.3.2) if the connectivity function $g(\mathbf{x}_j, \mathbf{x}_i)$ is set to connect all states within a small distance r of each other,

$$g(\mathbf{x}_i, \mathbf{x}_j) = g(\mathbf{x}_j, \mathbf{x}_i) = \begin{cases} 0 & d(\mathbf{x}_i, \mathbf{x}_j) > r \\ 1 & d(\mathbf{x}_i, \mathbf{x}_j) \leq r \end{cases}, \tag{2.16}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between states $\mathbf{x}_i$ and $\mathbf{x}_j$. In the limit as r goes to 0 (within an overall constant and scaling factor),

$$\begin{aligned} \lim_{r\to 0} K(\theta) &\sim K_{\mathrm{SM}}(\theta) \\ &= \sum_{\mathbf{x}\in\mathcal{D}}\left[\frac{1}{2}\nabla E(\mathbf{x})\cdot\nabla E(\mathbf{x}) - \nabla^2 E(\mathbf{x})\right], \end{aligned} \tag{2.17}$$

where $K_{\mathrm{SM}}(\theta)$ is the SM objective function. The full derivation is presented in Appendix D. Unlike SM, MPF is applicable to any parametric model, including discrete systems, and it does not require evaluating a third order derivative, which can result in unwieldy expressions.

2.8 Sampling the Connectivity Function g_{ij}

The MPF learning scheme is blind to regions in state space which are not directly connected via g_{ij} to data states. One way to more flexibly and thoroughly connect states is to treat g_{ij} as the probability of a connection from state j to state i, rather than as a binary indicator function. In this case, g_{ij} has the constraints required of a probability distribution,

$$\sum_i g_{ij} = 1 \tag{2.18}$$

$$g_{ij} \geq 0, \tag{2.19}$$

as well as the added constraint that if $g_{ij} > 0$ then $g_{ji} > 0$. Given these constraints for g_{ij}, the following form can be chosen for the transition rates Γ_{ij},

$$\Gamma_{ij}(\theta) = \begin{cases} g_{ij}\left(\frac{g_{ji}}{g_{ij}}\right)^{\frac{1}{2}} \exp\left[\frac{1}{2}\left(E_j(\theta) - E_i(\theta)\right)\right] & i \neq j \\ -\sum_{k \neq j} \Gamma_{kj}(\theta) & i = j \end{cases}. \tag{2.20}$$

It can be seen by substitution that the form for Γ_{ij} in Equation 2.20 still satisfies detailed balance. Additional motivations for this form are that Γ_{ij} have a linear factor g_{ij} so that a sum over i can be approximated using samples from g_{ij}, and that the contribution not included in the linear factor be a function solely of the ratio $\frac{g_{ji}}{g_{ij}}$, so that any (j independent) normalization term in g_{ij} cancels out.

Using Γ_{ij} from Equation 2.20, the MPF objective function becomes

$$K_{MPF}(\theta; \mathbf{g}) = \frac{1}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} \sum_{i \notin \mathcal{D}} g_{ij} \left(\frac{g_{ji}}{g_{ij}}\right)^{\frac{1}{2}} \exp\left[\frac{1}{2}\left(E_j(\theta) - E_i(\theta)\right)\right]. \tag{2.21}$$

This is identical to the original MPF objective function, except for the addition of a scaling term $\left(\frac{g_{ji}}{g_{ij}}\right)^{\frac{1}{2}}$ which compensates for the differences between the forward and backward connection probabilities g_{ij} and g_{ji}.

Because g_{ij} is a probability distribution, the inner sum in Equation 2.21 is an expectation over g_{ij}, and can be approximated by averaging over sample states i drawn from the distribution g_{ij}.

2.9 Persistent MPF

Recent work has shown that persistent particle techniques [Tieleman, 2008] outperform other sample driven learning techniques. In direct analogy to Persistent Contrastive Divergence

(PCD), and using the sampled connectivity function g_{ij} introduced in Section 2.8, MPF can be extended to perform learning with persistent particles.

Nearest neighbor schemes for setting the connectivity function $\mathbf{g}$ do not work nearly as well in continuous state spaces as in discrete state spaces, while Persistent MPF (PMPF) works quite well in continuous state spaces, so PMPF is particularly applicable to the continuous state space case.

2.9.1 Factoring K_{MPF}

In order to modify MPF to work with persistent samples, we first take advantage of a restricted form for g_{ij} to rewrite the MPF objective function. If the proposed connectivity function g_{ij} depends only on the destination state, i, and not the initial state, j, then the nested sums in Equation 2.21 can be factored apart. For the case that g_{ij} does not depend on j, we write it simply as g_i. The MPF objective function K_{MPF} becomes

$$K_{MPF}(\theta; \mathbf{g}) = \frac{1}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} \sum_{i \notin \mathcal{D}} g_i \left(\frac{g_j}{g_i} \right)^{\frac{1}{2}} \exp \left[\frac{1}{2} \left(E_j(\theta) - E_i(\theta) \right) \right] \tag{2.22}$$

$$= \left(\frac{1}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} \exp \left[\frac{1}{2} \left(E_j(\theta) + \log g_j \right) \right] \right) \cdot \left(\sum_{i \notin \mathcal{D}} g_i \exp \left[-\frac{1}{2} \left(E_i(\theta) + \log g_i \right) \right] \right). \tag{2.23}$$

The second sum is an expectation under g_i, and can be approximated by averaging over samples from g_i.

2.9.2 Iterative Improvement of g_i

The most informative states to connect to for learning are those which are most probable under the model distribution. Therefore, it is useful for learning to make g_i as similar to $p_i^{(\infty)}(\theta)$ as possible. An effective learning procedure alternates between updating g_i to resemble the current estimate of the model distribution $p_i^{(\infty)}\left(\hat{\theta}\right)$, and updating the estimated model parameters $\hat{\theta}$ using samples from a fixed connectivity function g_i. Defining a sequence of estimated parameter vectors $\hat{\theta}^n$ and proposed connectivity distributions g_i^n, where n indicates the learning iteration, this learning procedure becomes

1. Set $\hat{\theta}^0 =$ initial parameter guess
2. For $n \in \mathcal{Z}_+$ iterate

(a) Set $g_i^n = p_i^{(\infty)}\left(\hat{\theta}^{n-1}\right) = \frac{\exp\left[-E_i\left(\hat{\theta}^{n-1}\right)\right]}{Z\left(\hat{\theta}^{n-1}\right)}$

(b) Find $\hat{\theta}^n$ such that $K_{MPF}^n\left(\hat{\theta}^n\right) < K_{MPF}^n\left(\hat{\theta}^{n-1}\right)$

The MPF objective function at learning step n, $K_{MPF}^n(\theta)$, is written using the proposal distribution g_i^n set in step 2a,

$$K_{MPF}^n(\theta) = \left(\frac{1}{|\mathcal{D}|}\sum_{j\in\mathcal{D}}\exp\left[\frac{1}{2}\left(E_j(\theta) - E_j\left(\hat{\theta}^{n-1}\right)\right)\right]\right)\cdot \left(\sum_{i\notin\mathcal{D}} g_i^n \exp\left[-\frac{1}{2}\left(E_i(\theta) - E_i\left(\hat{\theta}^{n-1}\right)\right)\right]\right) \tag{2.24}$$

(the normalization terms in $\log g_i$ cancel out between the two sums). The expectation in the second sum is still evaluated using samples from $\mathbf{g}^n$. Typically, the number of samples drawn from $\mathbf{g}^n$ will be the same as the number of observations, $|\mathcal{D}|$.

2.9.3 Persistent Samples

The procedure in Section 2.9.2 will usually leave the proposal distribution at learning step n, $\mathbf{g}^n$, very similar to the proposal distribution from step $n-1$, $\mathbf{g}^{n-1}$. Significant time can thus be saved when generating samples from $\mathbf{g}^n$ by initializing with samples from $\mathbf{g}^{n-1}$, and taking only a small number of sampling steps.

2.9.4 Full Procedure for Persistent MPF

Using PMPF in an M-dimensional continuous states space $\mathcal{R}^M$, the parameter estimation procedure is as given in the steps below. $\mathcal{S}^n$ is the list of samples at learning step n. $|\mathcal{S}^n|$ is the number of samples - typically it will be the same as the number of observations $|\mathcal{D}|$.

1. Set $\hat{\theta}^0 =$ initial parameter guess
2. Initialize samples $\mathcal{S}^0$ (eg from a Gaussian)
3. For $n \in \mathcal{Z}_+$ iterate
   (a) Draw samples $\mathcal{S}^n$ from the distribution $p^{(\infty)}\left(\mathbf{x}; \hat{\theta}^{n-1}\right)$ via an MCMC sampler intialized at $\mathcal{S}^{n-1}$ (eg using Hamiltonian Monte Carlo)
   (b) Find $\hat{\theta}^n$ such that $K_{MPF}^n\left(\hat{\theta}^n\right) < K_{MPF}^n\left(\hat{\theta}^{n-1}\right)$ (eg via 10 steps of LBFGS gradient descent)

$K^n_{MPF}(\theta)$ is the MPF objective function at learning step n, and is written

$$K^n_{MPF}(\theta) = \left(\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \exp\left[\frac{1}{2}\left(E(\mathbf{x};\theta) - E\left(\mathbf{x};\hat{\theta}^{n-1}\right)\right)\right]\right) \cdot \left(\frac{1}{|\mathcal{S}^n|}\sum_{\mathbf{x}\in\mathcal{S}^n} \exp\left[-\frac{1}{2}\left(E(\mathbf{x};\theta) - E\left(\mathbf{x};\hat{\theta}^{n-1}\right)\right)\right]\right), \tag{2.25}$$

with derivative

$$\begin{aligned}\frac{\partial K^n_{MPF}(\theta)}{\partial\theta} =& \frac{1}{2}\left(\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \exp\left[\frac{1}{2}\left(E(\mathbf{x};\theta) - E\left(\mathbf{x};\hat{\theta}^{n-1}\right)\right)\right]\frac{\partial E(\mathbf{x};\theta)}{\partial\theta}\right) \cdot \\ & \left(\frac{1}{|\mathcal{S}^n|}\sum_{\mathbf{x}\in\mathcal{S}^n} \exp\left[-\frac{1}{2}\left(E(\mathbf{x};\theta) - E\left(\mathbf{x};\hat{\theta}^{n-1}\right)\right)\right]\right) \\ & - \frac{1}{2}\left(\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \exp\left[\frac{1}{2}\left(E(\mathbf{x};\theta) - E\left(\mathbf{x};\hat{\theta}^{n-1}\right)\right)\right]\right) \cdot \\ & \left(\frac{1}{|\mathcal{S}^n|}\sum_{\mathbf{x}\in\mathcal{S}^n} \exp\left[-\frac{1}{2}\left(E(\mathbf{x};\theta) - E\left(\mathbf{x};\hat{\theta}^{n-1}\right)\right)\right]\frac{\partial E(\mathbf{x};\theta)}{\partial\theta}\right).\end{aligned} \tag{2.26}$$

2.10 Summary

We have presented a novel, general purpose framework, called minimum probability flow learning (MPF), for parameter estimation in probabilistic models that outperforms current techniques in both learning time and accuracy. MPF works for any parametric model without hidden state variables, including those over both continuous and discrete state space systems, and it avoids explicit calculation of the partition function by employing deterministic dynamics in place of the slow sampling required by many existing approaches. Because MPF provides a simple and well-defined objective function, it can be minimized quickly using existing higher order gradient descent techniques. Furthermore, the objective function is convex for models in the exponential family, ensuring that the global minimum can be found with gradient descent in these cases. MPF was inspired by the minimum velocity approach developed by Movellan, and it reduces to that technique as well as to score matching and some forms of contrastive divergence for special cases of the dynamics.

Chapter 3

Minimum Probability Flow Experimental Results

In this chapter, we demonstrate experimentally the effectiveness of the Minimum Probability Flow (MPF) learning technique presented in Chapter 2. Matlab code implementing MPF for several of the cases presented in this chapter is available at [Sohl-Dickstein, 2010]. Unless stated otherwise, minimization was performed using the L-BFGS implementation in minFunc [Schmidt, 2005]. Material in this chapter is taken from [Hillar *et al.*, 2012b; Sohl-Dickstein *et al.*, 2011b; Sohl-Dickstein *et al.*, 2011a; Sohl-Dickstein *et al.*, 2009].

3.1 Ising Model

The Ising model [Ising, 1925] has a long and storied history in physics [Brush, 1967] and machine learning [Ackley *et al.*, 1985] and it has recently been found to be a surprisingly useful model for networks of neurons in the retina [Schneidman *et al.*, 2006; Shlens *et al.*, 2006]. The ability to fit Ising models to the activity of large groups of simultaneously recorded neurons is of current interest given the increasing number of these types of data sets from the retina, cortex and other brain structures.

3.1.1 Two Dimensional Ising Spin Glass

We estimated parameters for an Ising model (sometimes referred to as a fully visible Boltzmann machine or an Ising spin glass) of the form

$$p^{(\infty)}(\mathbf{x}; \mathbf{J}) = \frac{1}{Z(\mathbf{J})} \exp\left[-\mathbf{x}^{\mathrm{T}} \mathbf{J} \mathbf{x}\right], \tag{3.1}$$

where the coupling matrix $\mathbf{J}$ only had non-zero elements corresponding to nearest-neighbor units in a two-dimensional square lattice, and bias terms along the diagonal. The training data $\mathcal{D}$ consisted of $20,000$ d-element iid binary samples $\mathbf{x} \in \{0,1\}^d$ generated via Swendsen-Wang sampling [Swendsen and Wang, 1987] from a spin glass with known coupling parameters. We used a square 10×10 lattice, $d = 10^2$. The non-diagonal nearest-neighbor elements of $\mathbf{J}$ were set using draws from a normal distribution with variance $\sigma^2 = 10$. The diagonal (bias) elements of $\mathbf{J}$ were set in such a way that each column of $\mathbf{J}$ summed to 0, so that the expected unit activations were 0.5. The transition matrix $\mathbf{\Gamma}$ had $2^d \times 2^d$ elements, but for learning we populated it sparsely, setting

$$g_{ij} = g_{ji} = \begin{cases} 1 & \text{states } i,j \text{ differ by single bit flip} \\ 0 & \text{otherwise} \end{cases} . \tag{3.2}$$

The full derivation of the MPF objective for the case of an Ising model can be found in Appendix E.

Figure 3.1 shows the mean square error in the estimated $\mathbf{J}$ and the mean square error in the corresponding pairwise correlations as a function of learning time for MPF and four competing approaches: mean field theory with TAP corrections [Tanaka, 1998], CD with both one and ten sampling steps per iteration, and pseudolikelihood. Parameter estimation in Minimum Probability Flow and Pseudolikelihood was performed by applying an off the shelf L-BFGS (quasi-Newton gradient descent) implementation [Schmidt, 2005] to their objective functions evaluated over the full training dataset $\mathcal{D}$. CD was trained via stochastic gradient descent, using minibatches of size 100. The learning rate was annealed in a linear fashion from 3.0 to 0.1 to accelerate convergence. Mean field theory requires the computation of the inverse of the magnetic susceptibility matrix, which, for strong correlations, was often singular. A regularized pseudoinverse was used in the following manner:

$$A = (\chi^T \chi + \lambda I)^+ \chi^T, \tag{3.3}$$

where I is the identity matrix, M^+ denotes the Moore-Penrose pseudoinverse of a matrix M, χ is the magnetic susceptibility $\chi_{ij} = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle$, and λ is a regularizing parameter. This technique is known as stochastic robust approximation [Boyd and Vandenberghe, 2004].

Using MPF, learning took approximately 60 seconds, compared to roughly 800 seconds for pseudolikelihood and upwards of $20,000$ seconds for 1-step and 10-step CD. Note that given sufficient training samples, MPF would converge exactly to the right answer, as learning in the Ising model is convex (see Appendix B), and has its global minimum at the true solution. Table 3.1 shows the relative performance at convergence in terms of mean square error in recovered weights, mean square error in the resulting model's correlation function, and convergence time. MPF was dramatically faster to converge than any of the other models tested, with the exception of MFT+TAP, which failed to find reasonable parameters. MPF fit the model to the data substantially better than any of the other models.

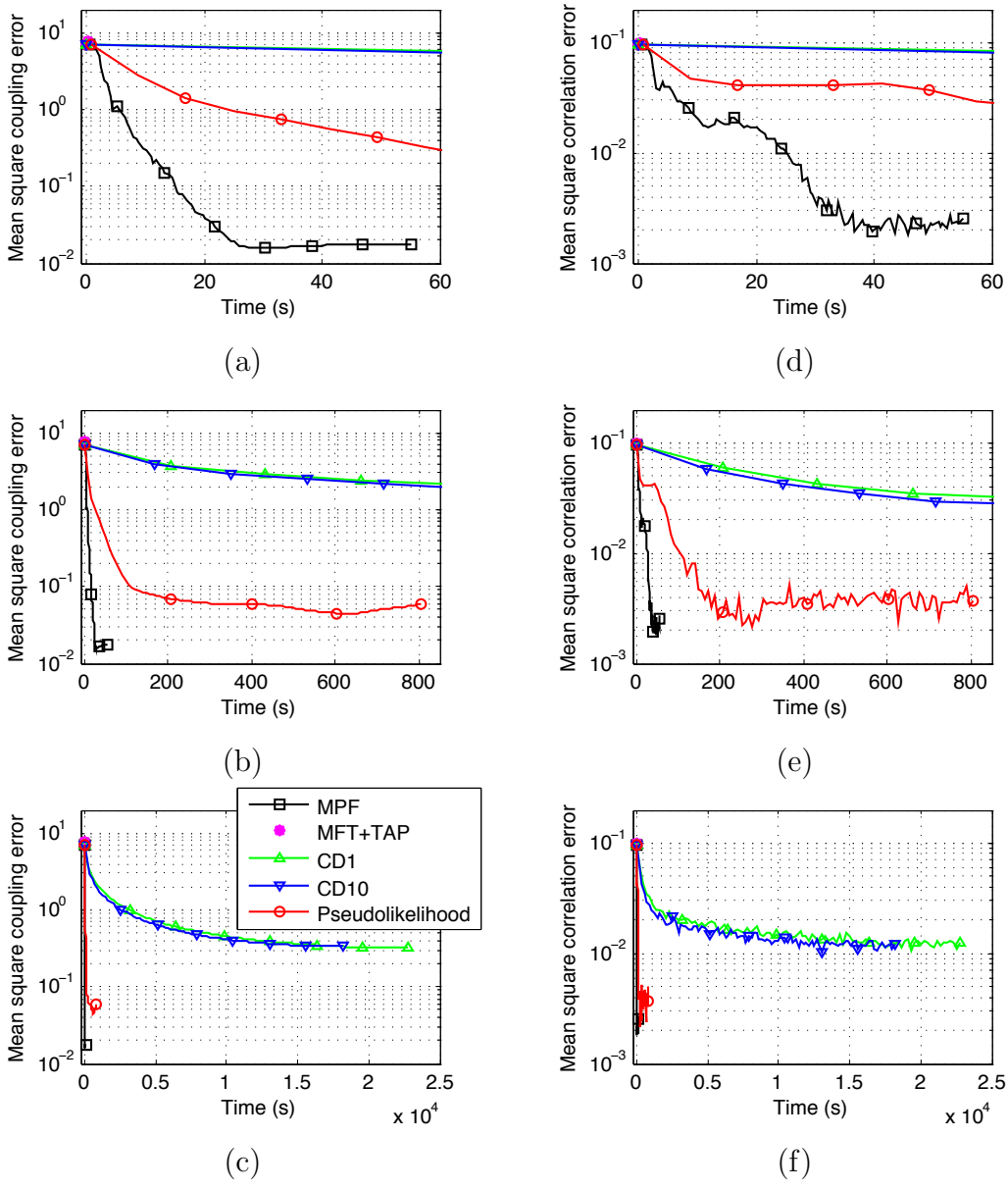


Figure 3.1: A demonstration of Minimum Probability Flow (MPF) outperforming existing techniques for parameter recovery in an Ising spin glass. **(a)** Time evolution of the mean square error in the coupling strengths for 5 methods for the first 60 seconds of learning. Note that mean field theory with second order corrections (MFT+TAP) actually increases the error above random parameter assignments in this case. **(b)** Mean square error in the coupling strengths for the first 800 seconds of learning. **(c)** Mean square error in coupling strengths for the entire learning period. **(d)–(f)** Mean square error in pairwise correlations for the first 60 seconds of learning, the first 800 seconds of learning, and the entire learning period, respectively. In every comparison above MPF finds a better fit, and for all cases but MFT+TAP does so in a shorter time (see Table 3.1).

Table 3.1: Mean square error in recovered coupling strengths (ϵ_J), mean square error in pairwise correlations (ϵ_{corr}) and learning time for MPF versus mean field theory with TAP correction (MFT+TAP), 1-step and 10-step contrastive divergence (CD-1 and CD-10), and pseudolikelihood (PL).

Technique	ϵ_J	ϵ_{corr}	Time (s)
MPF	0.0172	0.0025	∼60
MFT+TAP	7.7704	0.0983	0.1
CD-1	0.3196	0.0127	∼20000
CD-10	0.3341	0.0123	∼20000
PL	0.0582	0.0036	∼800

3.1.2 Fully Connected Ising Model Comparison

In order to allow an additional comparison to earlier work, we recovered the coupling parameters for the 40 unit, fully connected Ising model used in the 2008 paper "Faster solutions of the inverse pairwise Ising problem" [Broderick *et al.*, 2007]. Figure 3.2 shows the average error in predicted correlations as a function of learning time for 20,000 samples. The final absolute correlation error is 0.0058. The J_{ij} used were graciously provided by Broderick and coauthors, and were identical to those used for synthetic data generation in their paper [Broderick *et al.*, 2007]. Training was performed on 20,000 samples so as to match the number of samples used in section III.A. of Broderick et al. On an 8 core 2.33 GHz Intel Xeon, the learning converges in about 15 seconds. Broderick et al. perform a similar learning task on a 100-CPU grid computing cluster, with a convergence time of approximately 200 seconds.

3.2 Deep Belief Network

As a demonstration of learning on a more complex discrete valued model, we trained a 4 layer deep belief network (DBN) [Hinton *et al.*, 2006] on MNIST handwritten digits. A DBN consists of stacked restricted Boltzmann machines (RBMs), such that the hidden layer of one RBM forms the visible layer of the next. Each RBM has the form

$$p^{(\infty)}(\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{hid}}; \mathbf{W}) = \frac{\exp\left[\mathbf{x}_{\text{hid}}^T \mathbf{W} \mathbf{x}_{\text{vis}}\right]}{Z(\mathbf{W})}, \tag{3.4}$$

$$p^{(\infty)}(\mathbf{x}_{\text{vis}}; \mathbf{W}) = \frac{\exp\left[\sum_k \log\left(1 + \exp\left[\mathbf{W}_k \mathbf{x}_{\text{vis}}\right]\right)\right]}{Z(\mathbf{W})}. \tag{3.5}$$

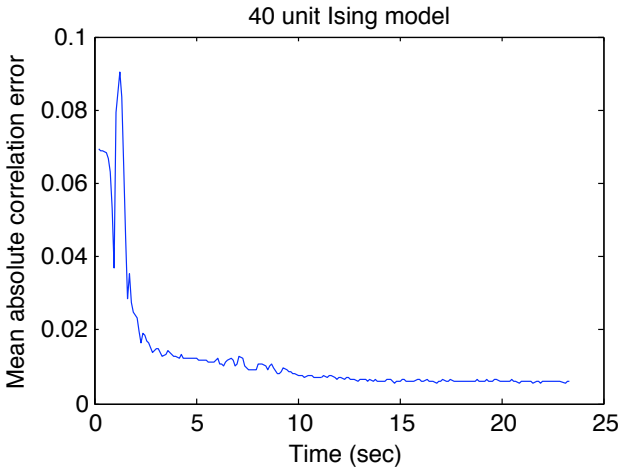


Figure 3.2: A demonstration of rapid fitting of a fully connected Ising model by minimum probability flow learning. The mean absolute error in the learned model's correlation matrix is shown as a functions of learning time for a 40 unit fully connected Ising model. Convergence is reached in about 15 seconds for $20,000$ samples.

Sampling-free application of MPF requires analytically marginalizing over the hidden units. RBMs were trained in sequence, starting at the bottom layer, on 10,000 samples from the MNIST postal hand written digits data set. As in the Ising case, the transition matrix $\mathbf{\Gamma}$ was populated so as to connect every state to all states that differed by only a single bit flip (Equation 3.2). The full derivation of the MPF objective for the case of an RBM can be found in Appendix F. Training was performed by both MPF and single step CD (note that CD turns into full ML learning as the number of steps is increased, and that many step CD would have produced a superior, more computationally expensive, answer).

Samples were generated by Gibbs sampling from the top layer RBM, then propagating each sample back down to the pixel layer by way of the conditional distribution $p^{(\infty)}(\mathbf{x}_{\text{vis}}|\mathbf{x}_{\text{hid}}; \mathbf{W}^k)$ for each of the intermediary RBMs, where k indexes the layer in the stack. $1,000$ sampling steps were taken between each sample. As shown in Figure 3.3, MPF learned a good model of handwritten digits.

3.3 Independent Component Analysis

As a demonstration of parameter estimation in continuous state space probabilistic models, we trained the receptive fields $\mathbf{J} \in R^{K \times K}$ of a K dimensional independent component analysis

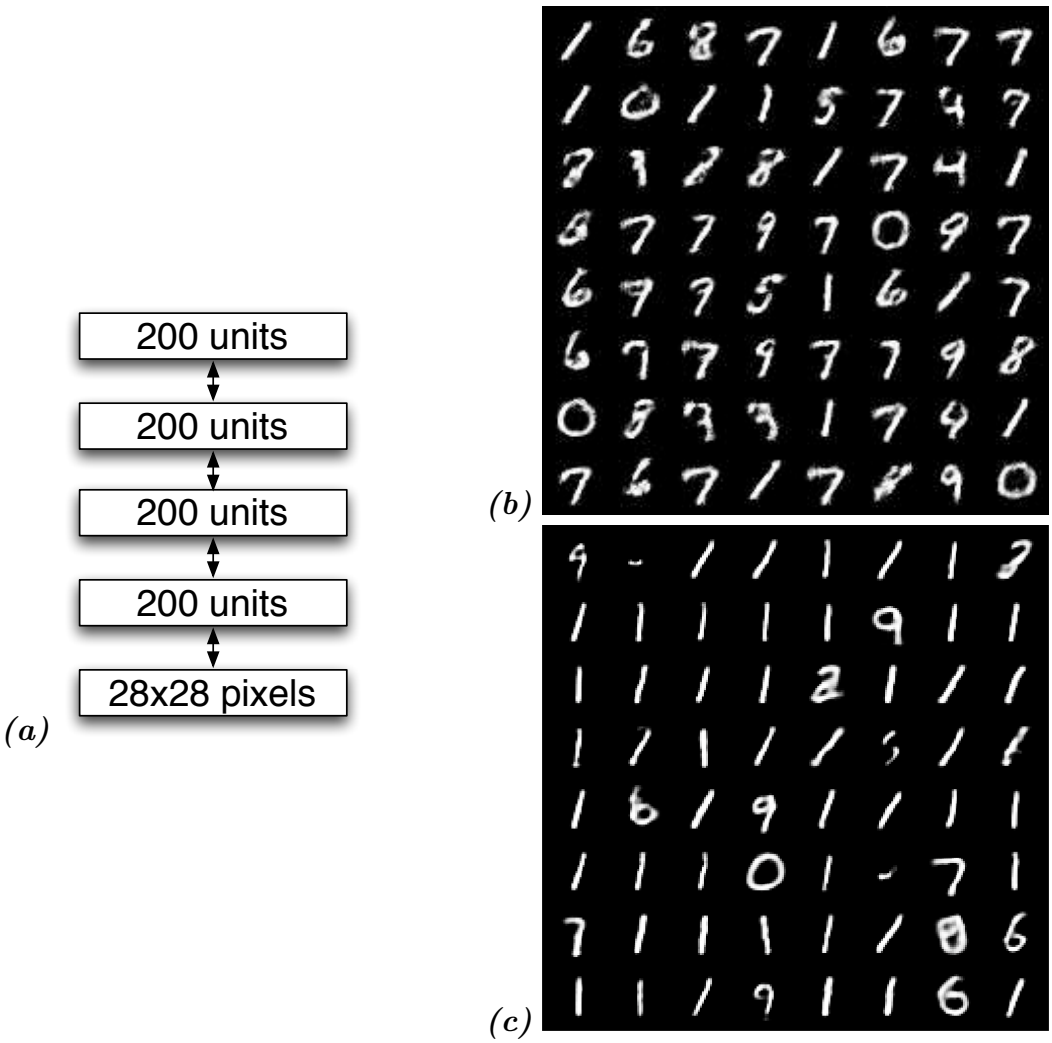
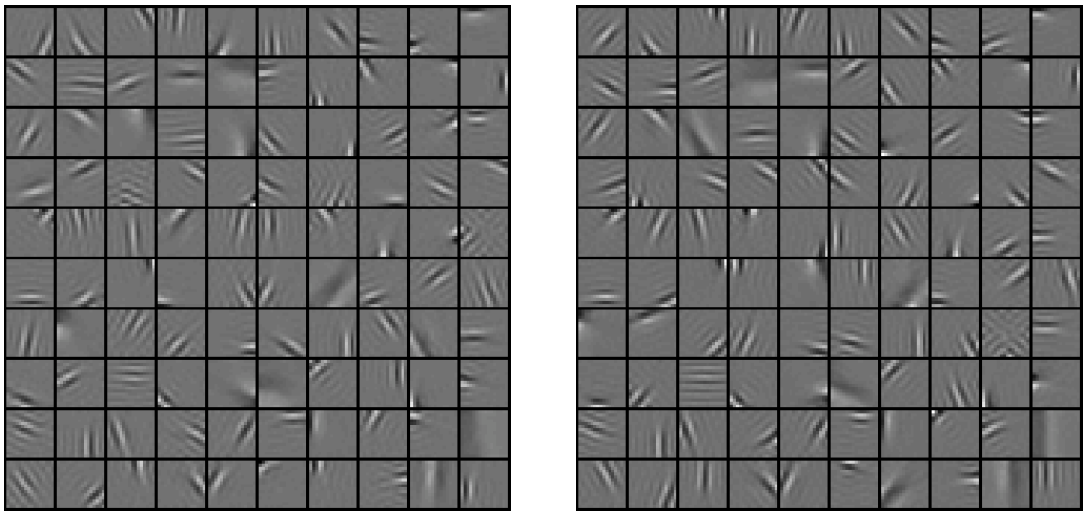


Figure 3.3: A deep belief network trained using minimum probability flow learning (MPF). *(a)* A four layer deep belief network was trained on the MNIST postal hand written digits dataset by MPF and single step contrastive divergence (CD). *(b)* Samples from the deep belief network after training via MPF. A reasonable probabilistic model for handwritten digits has been learned. *(c)* Samples after training via CD. The uneven distribution of digit occurrences suggests that CD-1 has learned a less representative model than MPF.



(a) *(b)*

Figure 3.4: A continuous state space model fit using minimum probability flow learning (MPF). Learned 10×10 pixel independent component analysis receptive fields $\mathbf{J}$ trained on natural image patches via *(a)* MPF and *(b)* maximum likelihood learning (ML). The average log likelihood of the model found by MPF (-120.61 nats) was nearly identical to that found by ML (-120.33 nats), consistent with the visual similarity of the receptive fields.

(ICA) [Bell AJ, 1995] model with a Laplace prior,

$$p^{(\infty)}(\mathbf{x};\mathbf{J}) = \frac{e^{-\sum_k |\mathbf{J}_k \mathbf{x}|}}{2^K \left|\mathbf{J}^{-1}\right|}, \tag{3.6}$$

on $100,000$ 10×10 whitened natural image patches from the van Hateren database [van Hateren and van der Schaaf, 1998]. Since the log likelihood and its gradient can be calculated analytically for ICA, we solved for $\mathbf{J}$ via both maximum likelihood learning and MPF, and compared the resulting log likelihoods. Both training techniques were initialized with identical Gaussian noise, and trained on the same data, which accounts for the similarity of individual receptive fields found by the two algorithms. The average log likelihood of the model after parameter estimation via MPF was -120.61 nats, while the average log likelihood after estimation via maximum likelihood was -120.33 nats. The receptive fields resulting from training under both techniques are shown in Figure 3.4. MPF parameter estimation was performed using the Persistent MPF (PMPF) algorithm described in Section 2.9, using Hamiltonian Monte Carlo (HMC) to sample from the connectivity function $g(\mathbf{x}_j, \mathbf{x}_i)$.

3.4 Memory Storage in a Hopfield Network

In 1982, motivated by the Ising spin glass model from statistical physics [Ising, 1925; Little, 1974], Hopfield introduced an auto-associative neural-network for the storage and retrieval of binary patterns [Hopfield, 1982]. Even today, this model and its various extensions [Cohen and Grossberg, 1983; Hinton and Sejnowski, 1986] provide a plausible mechanism for memory formation in the brain. However, existing techniques for training Hopfield networks suffer either from limited pattern capacity or excessive training time, and they exhibit poor performance when trained on unlabeled, corrupted memories.

In this section we show that MPF provides a tractable and neurally-plausible algorithm for the optimal storage of patterns in a Hopfield network, and we provide a proof that the capacity of such a network is at least one pattern per neuron. When compared with standard techniques for Hopfield pattern storage, MPF is shown to be superior in efficiency and generalization. Another finding is that MPF can store many patterns in a Hopfield network from highly corrupted (unlabeled) samples of them. This discovery is also corroborated visually by the storage of 64×64 binary images of human fingerprints from highly corrupted versions, as explained in Fig. 3.6.

3.4.1 Background

A *Hopfield network* $\mathcal{H} = (\mathbf{J}, \theta)$ on n nodes $\{1, \ldots, n\}$ consists of a symmetric *weight matrix* $\mathbf{J} = \mathbf{J}^\top \in \mathbb{R}^{n \times n}$ with zero diagonal and a *threshold vector* $\theta = (\theta_1, \ldots, \theta_n)^\top \in \mathbb{R}^n$. The possible *states* of the network are all length n binary strings $\{0,1\}^n$, which we represent as

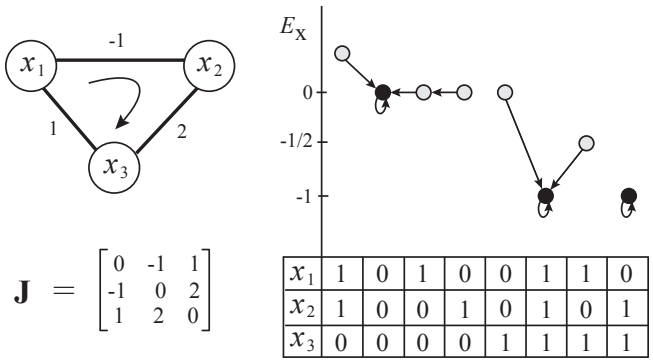


Figure 3.5: **Example Hopfield Network.** The figure above displays a 3-node Hopfield network with weight matrix $\mathbf{J}$ and zero threshold vector. Each binary state vector $\mathbf{x} = (x_1, x_2, x_3)^\top$ has energy $E_\mathbf{x}$ as labeled on the y-axis of the diagram on the right. Arrows between states represent one iteration of the network dynamics; i.e., x_1, x_2, and x_3 are updated by (3.7) in the order indicated by the clockwise arrow in the graph on the left. The resulting fixed states of the network are indicated by filled circles.

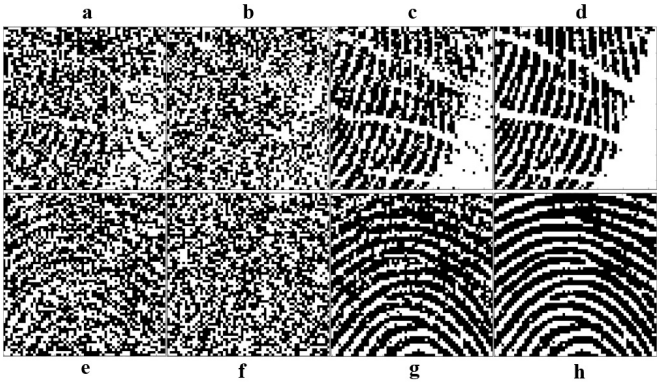


Figure 3.6: **Learning memories from corrupted samples.** We stored 80 fingerprints (64×64 binary images) in a Hopfield network with $n = 64^2 = 4096$ nodes by minimizing the MPF objective (3.10) over a large set of randomly generated (and unlabeled) "noisy" versions (each training pattern had a random subset of 1228 of its bits flipped; e.g., a,e). After training, all 80 original fingerprints were stored as fixed-points of the network. **a.** Sample fingerprint with 30% corruption used for training. **b.** Sample fingerprint with 40% corruption. **c.** State of the network after one update of the dynamics initialized at b. **d.** Converged network dynamics equal to original fingerprint. **e-h.** As in a-d, but for a different fingerprint.

binary column vectors $\mathbf{x} = (x_1, \ldots, x_n)^\top$, each $x_i \in \{0, 1\}$ indicating the state x_i of node i. Given any state $\mathbf{x} = (x_1, \ldots, x_n)^\top$, an (asynchronous) *dynamical update* of $\mathbf{x}$ consists of

replacing x_i in $\mathbf{x}$ (in consecutive order starting with $i = 1$; see Fig 3.5) with the value

$$x_i = H(\mathbf{J}_i\mathbf{x} - \theta_i). \tag{3.7}$$

Here, $\mathbf{J}_i$ is the ith row of $\mathbf{J}$ and H is the *Heaviside function* given by $H(r) = 1$ if $r > 0$ and $H(r) = 0$ if $r \leq 0$.

The *energy* $E_\mathbf{x}$ of a binary pattern $\mathbf{x}$ in a Hopfield network is defined to be

$$E_\mathbf{x}(\mathbf{J}, \theta) := -\frac{1}{2}\mathbf{x}^\top\mathbf{J}\mathbf{x} + \theta^\top\mathbf{x} = -\sum_{i<j} x_i x_j J_{ij} + \sum_{i=1}^{n} \theta_i x_i, \tag{3.8}$$

identical to the energy function for an Ising spin glass. In fact, the dynamics of a Hopfield network can be seen as 0-temperature Gibbs sampling of this energy function. A fundamental property of Hopfield networks is that asynchronous dynamical updates do not increase the energy (3.8). Thus, after a finite number of updates, each initial state $\mathbf{x}$ converges to a *fixed-point* $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)^\top$ of the dynamics; that is, $x_i^* = H(\mathbf{J}_i\mathbf{x}^* - \theta_i)$ for each i. See Fig. 3.5 for a sample Hopfield network on $n = 3$ nodes.

Given a binary pattern $\mathbf{x}$, the *neighborhood* $\mathcal{N}(\mathbf{x})$ of $\mathbf{x}$ consists of those binary vectors which are Hamming distance 1 away from $\mathbf{x}$ (i.e., those with exactly one bit different from $\mathbf{x}$). We say that $\mathbf{x}$ is a *strict local minimum* if every $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ has a strictly larger energy:

$$0 > E_\mathbf{x} - E_{\mathbf{x}'} = (\mathbf{J}_i\mathbf{x} - \theta_i)\delta_i, \tag{3.9}$$

where $\delta_i = 1 - 2x_i$ and x_i is the bit that differs between $\mathbf{x}$ and $\mathbf{x}'$. It is straightforward to verify that if $\mathbf{x}$ is a strict local minimum, then it is a fixed-point of the dynamics.

A basic problem is to construct Hopfield networks with a given set $\mathcal{D}$ of binary patterns as fixed-points or strict local minima of the energy function (3.8). Such networks are useful for memory denoising and retrieval since corrupted versions of patterns in $\mathcal{D}$ will converge through the dynamics to the originals. Traditional approaches to this problem consist of iterating over $\mathcal{D}$ a *learning rule* [Hertz *et al.*, 1991] that updates a network's weights and thresholds given a training pattern $\mathbf{x} \in \mathcal{D}$. We call a rule *local* when the learning updates to the three parameters J_{ij}, θ_i, and θ_j can be computed with access solely to x_i, x_j, the feedforward inputs $\mathbf{J}_i\mathbf{x}$, $\mathbf{J}_j\mathbf{x}$, and the thresholds θ_i, θ_j; otherwise, we call the rule *nonlocal*. Note that a stricter definition of local is sometimes used, in which a learning rule is called local only if updating J_{ij} depends on the states x_i and x_j, but not on the feedforward inputs to units i and j. Each unit i necessarily has its feedforward input $\mathbf{J}_i\mathbf{x}$ locally available, since the feedforward input is compared against the threshold θ_i when the output x_i is chosen. We therefore label learning rules which utilize feedforward input as local rules. The locality of a rule is an important feature in a network training algorithm because of its necessity in theoretical models of computation in neuroscience.

In [Hopfield, 1982], Hopfield defined an *outer-product learning rule* (OPR) for finding such networks. OPR is a local rule since only the binary states of nodes x_i and x_j are required to update a coupling term J_{ij} during training (and only the state of x_i is required to update θ_i). Using OPR, at most $n/(4 \log n)$ patterns can be stored without errors in an n-node Hopfield network [Weisbuch and Fogelman-Soulié, 1985; McEliece *et al.*, 1987]. In particular, the ratio of patterns storable to the number of nodes using this rule is at most $1/(4 \log n)$ memories per neuron, which approaches zero as n increases. If a small percentage of incorrect bits is tolerated, then approximately $0.15n$ patterns can be stored [Hopfield, 1982; Amit *et al.*, 1987].

The *perceptron learning rule* (PER) [Rosenblatt, 1957; Minsky and Papert, 1988] provides an alternative method to store patterns in a Hopfield network [Jinwen, 1993]. PER is also a local rule since updating J_{ij} requires only $\mathbf{J}_i\mathbf{x}$ and $\mathbf{J}_j\mathbf{x}$ (and updating θ_i requires $\mathbf{J}_i\mathbf{x}$). Unlike OPR, it achieves optimal storage capacity, in that if it is possible for a collection of patterns $\mathcal{D}$ to be fixed-points of a Hopfield network, then PER will converge to parameters $\mathbf{J}, \theta$ for which all of $\mathcal{D}$ are fixed-points. However, training frequently takes many parameter update steps (see Fig. 3.8), and the resulting Hopfield networks do not generalize well (see Fig. 3.9) nor store patterns from corrupted samples (see Fig. 3.10).

Despite the connection to the Ising model energy function, and the common usage of Ising spin glasses (otherwise referred to as Boltzmann machines [Hinton and Sejnowski, 1986]) to build probabilistic models of binary data, we are aware of no previous work on associative memories that takes advantage of a probabilistic interpretation during training. Probabilistic interpretations have been used for pattern recovery [Sommer and Dayan, 1998].

3.4.2 Theoretical Results

We give an efficient algorithm for storing at least n binary patterns as strict local minima (and thus fixed-points) in an n-node Hopfield network, and we prove that this algorithm achieves the optimal storage capacity achievable in such a network. We also present a novel local learning rule for the training of neural networks.

Consider a collection of m binary n-bit patterns $\mathcal{D}$ to be stored as strict local minima in a Hopfield network. Not all collections of m such patterns $\mathcal{D}$ can so be stored; for instance, from (3.9) we see that no two binary patterns one bit apart can be stored simultaneously. Nevertheless, we say that the collection $\mathcal{D}$ *can be stored as local minima* of a Hopfield network if there is some $\mathcal{H} = (\mathbf{J}, \theta)$ such that each $\mathbf{x} \in \mathcal{D}$ is a strict local minimum of the energy function $E_{\mathbf{x}}(\mathbf{J}, \theta)$ in (3.8).

The *minimum probability flow* (MPF) objective function given the collection $\mathcal{D}$ is

$$K_{\mathcal{D}}(\mathbf{J}, \theta) := \sum_{\mathbf{x} \in \mathcal{D}} \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \exp\left(\frac{E_{\mathbf{x}} - E_{\mathbf{x}'}}{2}\right). \tag{3.10}$$

The function in (3.10) is infinitely differentiable and strictly convex in the parameters. Notice that when $K_{\mathcal{D}}(\mathbf{J}, \theta)$ is small, the energy differences $E_{\mathbf{x}} - E_{\mathbf{x}'}$ between $\mathbf{x} \in \mathcal{D}$ and patterns $\mathbf{x}'$ in neighborhoods $\mathcal{N}(\mathbf{x})$ will satisfy (3.9), making $\mathbf{x}$ a fixed-point of the dynamics.

As the following result explains, minimizing (3.10) given a storable set of patterns will determine a Hopfield network storing those patterns.

Theorem 1. *If a set of binary vectors $\mathcal{D}$ can be stored as local minima of a Hopfield network, then minimizing the convex MPF objective (3.10) will find such a network.*

Proof: We first claim that $\mathcal{D}$ can be stored as local minima of a Hopfield network $\mathcal{H}$ if and only if the MPF objective (3.10) satisfies $K_{\mathcal{D}}(\mathbf{J}, \theta) < 1$ for some $\mathbf{J}$ and θ. Suppose first that $\mathcal{D}$ can be made strict local minima with parameters $\mathbf{J}$ and θ. Then for each $\mathbf{x} \in \mathcal{D}$ and $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$, inequality (3.9) holds. In particular, a uniform scaling in the parameters will make the energy differences in (3.10) arbitrarily large and negative, and thus K can be made less than 1. Conversely, suppose that $K_{\mathcal{D}}(\mathbf{J}, \theta) < 1$ for some choice of $\mathbf{J}$ and θ. Then each term in the sum of positive numbers (3.10) is less than 1. This implies that the energy difference between each $\mathbf{x} \in \mathcal{D}$ and $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ satisfies (3.9). Thus, $\mathcal{D}$ are all strict local minima.

We now explain how the claim proves the theorem. Suppose that $\mathcal{D}$ can be stored as local minima of a Hopfield network; then, $K_{\mathcal{D}}(\mathbf{J}, \theta) < 1$ for some $\mathbf{J}, \theta$. Any method producing parameter values $\mathbf{J}$ and θ having objective (3.10) arbitrarily close to the infimum of $K_{\mathcal{D}}(\mathbf{J}, \theta)$ will produce a network with MPF objective strictly less than 1, and therefore store $\mathcal{D}$ by above. □

Our next main result is that at least n patterns in an n-node Hopfield network can be stored by minimizing (3.10). To make this statement mathematically precise, we introduce some notation. Let $r(m, n) < 1$ be the probability that a collection of m binary patterns chosen uniformly at random from all $\binom{2^n}{m}$ m-element subsets of $\{0, 1\}^n$ can be made local minima of a Hopfield network. The *pattern capacity* (per neuron) of the Hopfield network is defined to be the supremum of all real numbers $a > 0$ such that

$$\lim_{n \to \infty} r(an, n) = 1. \tag{3.11}$$

Theorem 2. *The pattern capacity of an n-node Hopfield network is at least 1 pattern per neuron.*

In other words, for any fixed $a < 1$, the fraction of all subsets of $m = an$ patterns that can be made strict local minima (and thus fixed-points) of a Hopfield network with n nodes converges to 1 as n tends to infinity. Moreover, by Theorem 1, such networks can be found by minimizing (3.10). Although the Cover bound [Cover, 1965] forces $a \leq 2$, it is an open problem to determine the exact critical value of a (i.e., the exact pattern capacity of the Hopfield network). Note that a perceptron with an asymmetric weight matrix can achieve the Cover bound and store $2N$ arbitrary mappings, but its stored mappings will not be local minima of an associated energy function, and the learned network will not be equivalent to

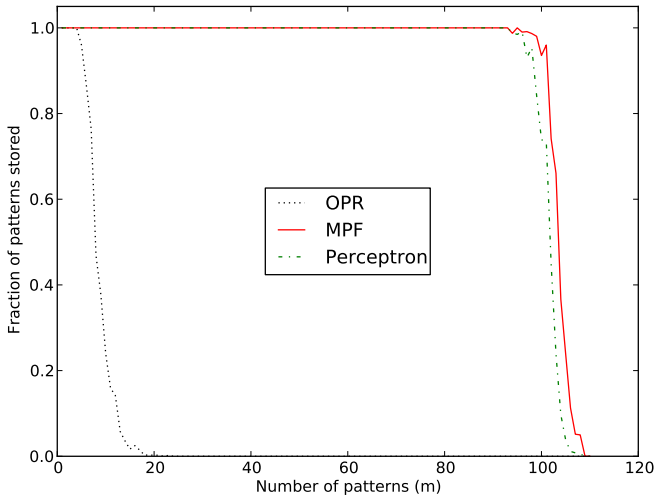


Figure 3.7: Shows fraction of patterns made fixed-points of a Hopfield network using OPR (outer-product rule), MPF (minimum probability flow), and PER (perceptron) as a function of the number of randomly generated training patterns m. Here, $n = 64$ binary nodes and we have averaged over $t = 20$ trials. The slight difference in performance between MPF and PER is due to the extraordinary number of iterations required for PER to achieve perfect storage of patterns near the critical pattern capacity of the Hopfield network. See also Fig. 3.8.

a Hopfield network [Gardner, 1987]. Experimental evidence suggests that the limit in (3.11) is 1 for all $a < 1.5$, but converges to 0 for $a > 1.7$ (see Fig. 3.7).

We close this section by defining a new learning rule for a neural network. In words, the *minimum probability flow learning rule* (MPF) takes an input training pattern $\mathbf{x}$ and moves the parameters $(\mathbf{J}, \theta)$ a small amount in the direction of steepest descent of the MPF objective function $K_{\mathcal{D}}(\mathbf{J}, \theta)$ with $\mathcal{D} = \{\mathbf{x}\}$. Mathematically, these updates for J_{ij} and θ_i take the form (where again, $\delta = \mathbf{1} - 2\mathbf{x}$):

$$\Delta J_{ij} \propto -\delta_i x_j e^{\frac{1}{2}(\mathbf{J}_i\mathbf{x} - \theta_i)\delta_i} - \delta_j x_i e^{\frac{1}{2}(\mathbf{J}_j\mathbf{x} - \theta_j)\delta_j} \tag{3.12}$$

$$\Delta\theta_i \propto \delta_i e^{\frac{1}{2}(\mathbf{J}_i\mathbf{x} - \theta_i)\delta_i}. \tag{3.13}$$

It is clear from (3.12),(3.13) that MPF is a local learning rule.

3.4.3 Experimental Results

We performed several experiments comparing standard techniques for fitting Hopfield networks with minimizing the MPF objective function (3.10). All computations were performed on standard desktop computers, and we used used the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [Nocedal, 1980] to minimize (3.10).

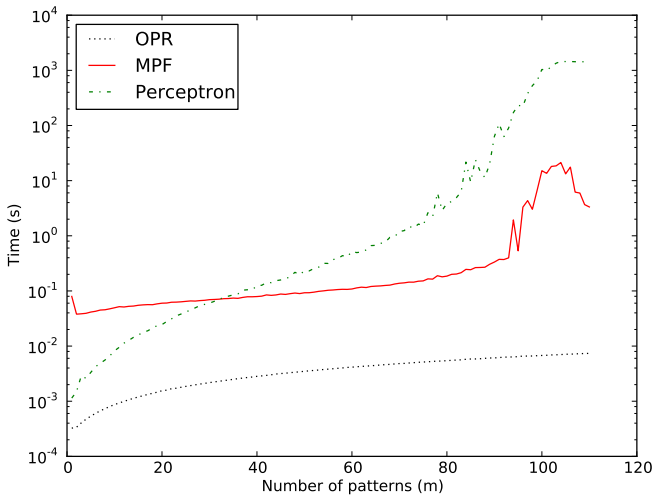


Figure 3.8: Shows time (on a log scale) to train a Hopfield network with $n = 64$ neurons to store m patterns using OPR, PER, and MPF (averaged over $t = 20$ trials).

In our first experiment, we compared MPF to the two methods OPR and PER for finding 64-node Hopfield networks storing a given set of patterns $\mathcal{D}$. For each of 20 trials, we used the three techniques to store a randomly generated set of m binary patterns, where m ranged from 1 to 120. The results are displayed in Fig. 3.7 and support the conclusions of Theorem 1 and Theorem 2.

To study the efficiency of our method, we compared training time of a 64-node network as in Fig. 3.7 with the three techniques OPR, MPF, and PER. The resulting computation times are displayed in Fig. 3.8 on a logarithmic scale. Notice that computation time for MPF and PER significantly increases near the pattern capacity threshold of the Hopfield network.

For our third experiment, we compared the denoising performance of MPF and PER. For each of four values for m in a 128-node Hopfield network, we determined weights and thresholds for storing all of a set of m randomly generated binary patterns using both MPF and PER. We then flipped 0 to 64 of the bits in the stored patterns and let the dynamics (3.7) converge (with weights and thresholds given by MPF and PER), recording if the converged pattern was identical to the original pattern or not. Our results are shown in Fig 3.9, and they demonstrate the superior corrupted memory retrieval performance of MPF.

A surprising final finding in our investigation was that MPF can store patterns from highly corrupted or noisy versions on its own and without supervision. This result is explained in Fig 3.10. To illustrate the experiment visually, we stored $m = 80$ binary fingerprints in a 4096-node Hopfield network using a large set of training samples which were corrupted by flipping at random 30% of the original bits; see Fig. 3.6 for more details.

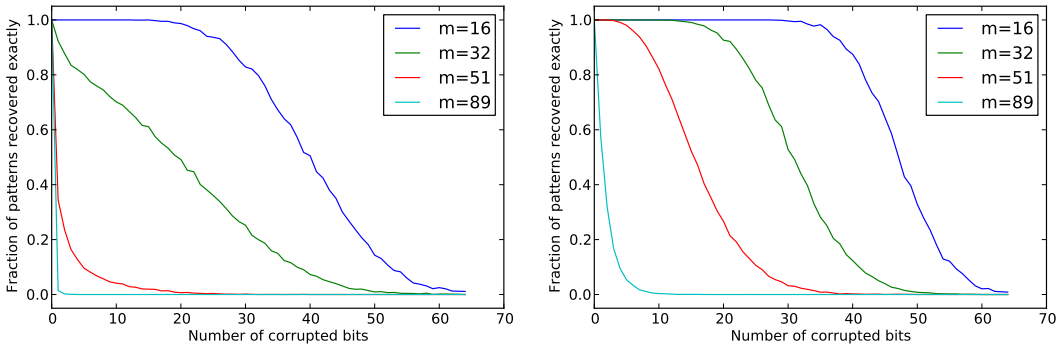


Figure 3.9: Shows fraction of exact pattern recovery for a perfectly trained $n = 128$ Hopfield network using rules PER (figure on the left) and MPF (figure on the right) as a function of bit corruption at start of recovery dynamics for various numbers m of patterns to store. We remark that this figure and the next do not include OPR as its performance was far worse than either MPF or PER.

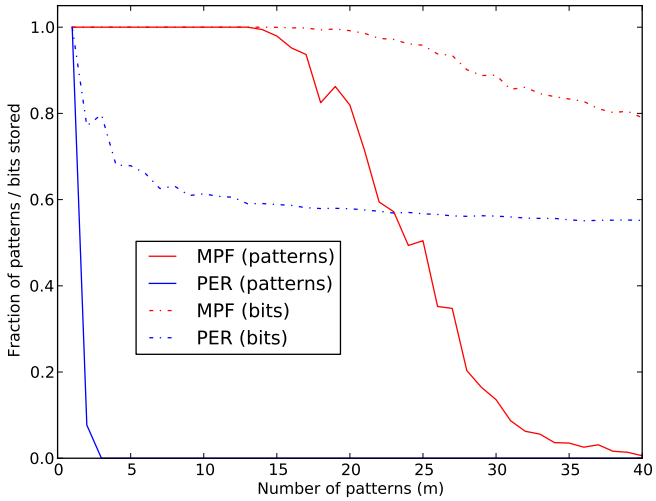


Figure 3.10: Shows fraction of patterns (shown in red for MPF and blue for PER) and fraction of bits (shown in dotted red for MPF and dotted blue for PER) recalled of trained networks (with $n = 64$ nodes each) as a function of the number of patterns m to be stored. Training patterns were presented repeatedly with 20 bit corruption (i.e., 31% of the bits flipped). (averaged over t = 13 trials.)

3.4.4 Discussion

We have presented a novel technique for the storage of patterns in a Hopfield associative memory. The first step of the method is to fit an Ising model using minimum probability flow learning to a discrete distribution supported equally on a set of binary target patterns. Next, we use the learned Ising model parameters to define a Hopfield network. We show

that when the set of target patterns is storable, these steps result in a Hopfield network that stores all of the patterns as fixed-points. We have also demonstrated that the resulting (convex) algorithm outperforms current techniques for training Hopfield networks.

We have shown improved recovery of memories from noisy patterns and improved training speed as compared to training by PER. We have demonstrated optimal storage capacity in the noiseless case, outperforming OPR. We have also demonstrated the unsupervised storage of memories from heavily corrupted training data. Furthermore, the learning rule that results from our method is local; that is, updating the weights between two units requires only their states and feedforward input.

It is the probabilistic interpretation of the Hopfield network used in the MPF rule that leads to the superior robustness to noise and graceful degradation in the case of more patterns than can be stored as fixed points. The probabilistic learning objective tries not only to make the observed data states probable, but also to make unobserved states improbable. This second aspect reduces the probability mass assigned to spurious minima and their attractive wells, improving pattern recovery from noisy initialization. Additionally, when more patterns are presented than can be stored, the probabilistic objective attempts to carve out a broad minima in the energy landscape around clusters of datapoints. If many noisy examples of template patterns are presented, the lowest energy states will tend to lie in the center of the minima corresponding to each cluster of data, and will thus tend to correspond to the template states.

As MPF allows the fitting of large Hopfield networks quickly, new investigations into the structure of Hopfield networks are posssible [Hillar *et al.*, 2012a]. It is our hope that the robustness and speed of this learning technique will enable practical use of Hopfield associative memories in both computational neuroscience, computer science, and scientific modeling.

Chapter 4

The Natural Gradient by Analogy to Signal Whitening, and Recipes and Tricks for its Use

Difficulties in training probabilistic models can stem from ill conditioning of the model's parameter space as well as from an inability to analytically normalize the model. In this chapter we review how an ill conditioned parameter space can undermine learning, and we present a novel interpretation of a common technique for dealing with this ill conditioning, the natural gradient. In addition, we present tricks and specific prescriptions for applying the natural gradient to learning problems. Material in this chapter is taken from [Sohl-Dickstein, 2012b].

The natural gradient, as introduced by [Amari, 1987], allows for more efficient gradient descent by removing dependencies and biases inherent in a function's parameterization. Several papers present the topic thoroughly and precisely [Amari, 1987; Amari, 1998; Amari and Nagaoka, 2000; Theis, 2005; Amari, 2010]. It remains a very difficult idea to get your head around however. The intent of this chapter is to provide simple intuition for the natural gradient and its uses. The natural gradient is explained by analogy to the more widely understood concept of signal whitening. To our knowledge, this is the first time a connection has been made between signal whitening and the natural gradient.

4.1 Natural gradient

4.1.1 A simple example

We begin with a simple probabilistic model which has clearly been very poorly parametrized. For this we use a two dimensional gaussian distribution, with means written in terms of the

parameters $\theta \in \mathcal{R}^2$,

$$q(\mathbf{x};\theta) = \frac{1}{2\pi}\exp\left[-\frac{1}{2}\left(x_1 - \left[3\theta_1 + \frac{1}{3}\theta_2\right]\right)^2 - \frac{1}{2}\left(x_2 - \left[\frac{1}{3}\theta_1\right]\right)^2\right]. \tag{4.1}$$

As an objective function $J(\theta)$ we use the negative log likelihood of $q(\mathbf{x};\theta)$ under an observed data distribution $p(\mathbf{x})$

$$J(\theta) = -\left\langle \log q(\mathbf{x};\theta)\right\rangle_{p(\mathbf{x})}. \tag{4.2}$$

Using steepest gradient descent to minimize the negative log likelihood involves taking steps like

$$\Delta\theta \propto -\nabla_\theta J(\theta) \tag{4.3}$$

$$\begin{bmatrix}\Delta\theta_1 \\ \Delta\theta_2\end{bmatrix} \propto \begin{bmatrix}\left\langle 3\left(x_1 - \left[3\theta_1 + \frac{1}{3}\theta_2\right]\right) + \frac{1}{3}\left(x_2 - \left[\frac{1}{3}\theta_1\right]\right)\right\rangle_{p(\mathbf{x})} \\ \left\langle \frac{1}{3}\left(x_1 - \left[3\theta_1 + \frac{1}{3}\theta_2\right]\right)\right\rangle_{p(\mathbf{x})}\end{bmatrix}. \tag{4.4}$$

As can be seen in Figure 4.1*a* the steepest gradient update steps can move the parameters in a direction nearly perpendicular to the desired direction. $q(\mathbf{x};\theta)$ is much more sensitive to changes in θ_1 than θ_2, so the step size in θ_1 should be much smaller, but is instead much larger. In addition, θ_1 and θ_2 are not independent of each other. They move the distribution in nearly the same direction, making movement in the perpendicular direction particularly difficult. Getting the parameters here to fully converge via steepest descent is a slow proposition, as shown in Figure 4.1*b*.

The pathological learning gradient above is illustrative of a more general problem. A model's learning gradient is effected by the parameterization of the model as well as the objective function being minimized. The effects of the parameterization can dominate learning. The natural gradient is a technique to remove the effects of model parameterization from learning updates.

4.1.2 A metric on the parameter space

As a first step towards compensating for differences in relative scaling, and cross-parameter dependencies, the shape of the parameter space θ is first described by assigning it a measure of distance, or a metric. This metric is expressed via a symmetric matrix $\mathbf{G}(\theta)$, which defines the length $|d\theta|$ of an infinitesimal step $d\theta$ in the parameters,

$$|d\theta|^2 = \sum_i \sum_j G_{ij}(\theta)\, d\theta_i d\theta_j = d\theta^T \mathbf{G}(\theta)\, d\theta. \tag{4.5}$$

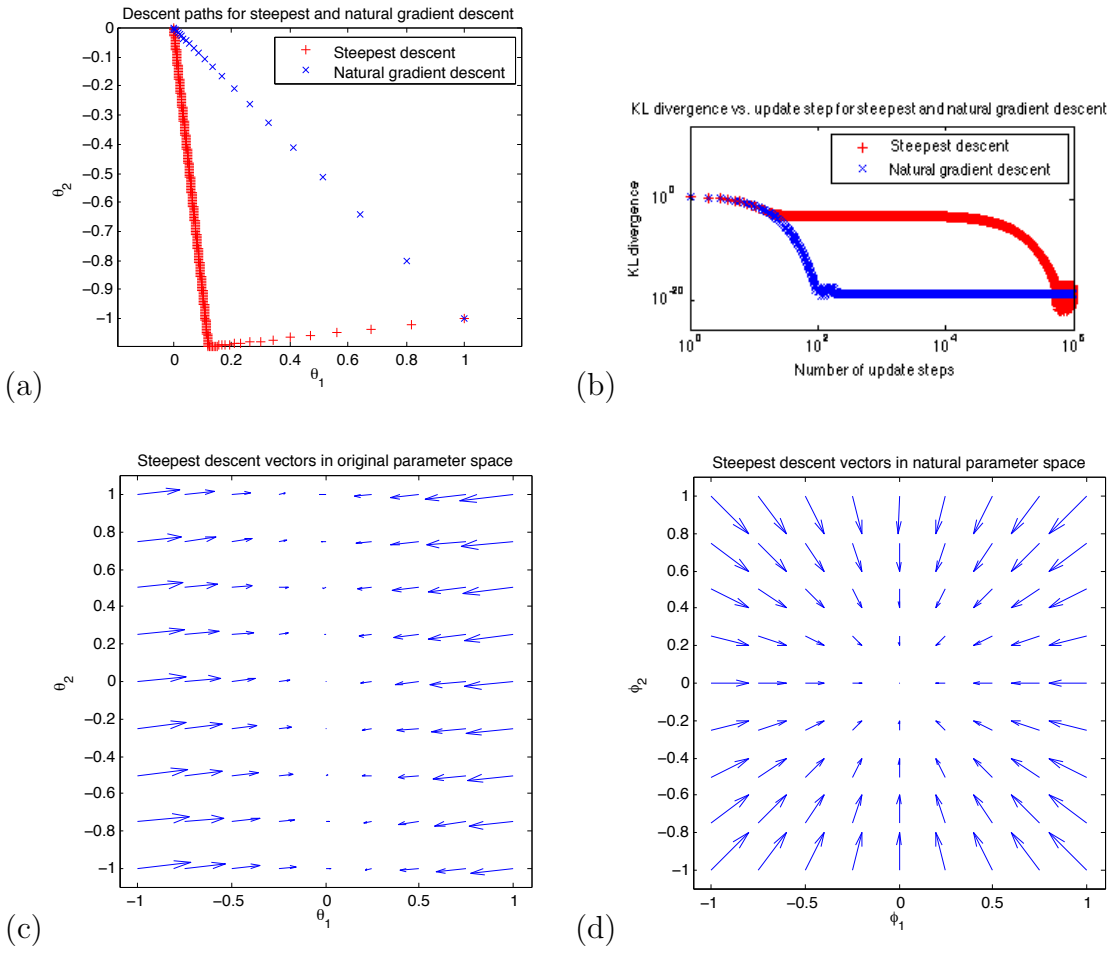


Figure 4.1: *(a)* The parameter descent paths taken by steepest gradient descent (red) and natural gradient descent (blue) for the example given in Section 4.1.1. The parameters are initialized at $\theta_{init} = [1, -1]^T$, and are fit to data generated with $\theta_{true} = [0, 0]^T$. The Fisher information matrix (Equation 4.30) is used to calculate the natural gradient. Notice that steepest descent takes a more circuitous and far slower path. *(b)* The KL divergence between the data distribution and the fit model as a function of number of gradient descent steps. Descent using the natural gradient converges more quickly. *(c)* The arrows give the gradient of the log likelihood objective (Equation 4.2), for a grid of parameter settings. This is the descent direction provided by Equation 4.4. *(d)* The gradient of the same log likelihood objective (Equation 4.2), but in terms of the whitened, natural, parameter space ϕ as described in Section 4.1.4. Note that steepest descent in the whitened space converges directly to the true parameter values $\phi_{true} = \mathbf{G}^{\frac{1}{2}}\theta_{true} = [0, 0]^T$.

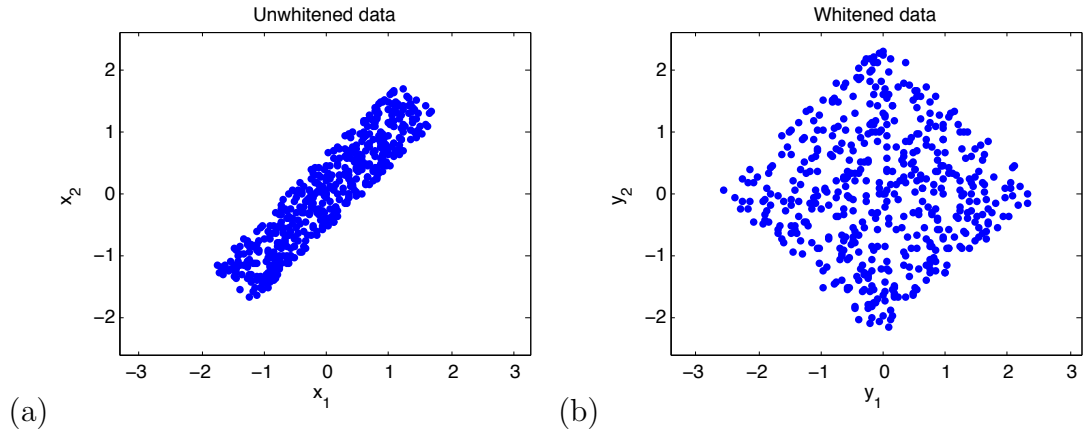


Figure 4.2: Example of signal whitening. *(a)* Samples $\mathbf{x}$ from an unwhitened distribution in 2 variables. *(b)* The same samples after whitening, in new variables $\mathbf{y} = \mathbf{W}\mathbf{x} = \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{x}$.

$\mathbf{G}(\theta)$ is chosen so that the length $|d\theta|$ provides a reasonable measure for the expected magnitude of the difference of $J(\theta + d\theta)$ from $J(\theta)$. That is, $\mathbf{G}(\theta)$ is chosen such that $|d\theta|$ is representative of the expected magnitude of the change in the objective function resulting from a step $d\theta$. There is no uniquely correct choice for $\mathbf{G}(\theta)$.

If the objective function $J(\theta)$ is the log likelihood of a probability distribution $q(\mathbf{x};\theta)$, then a measure of the information distance between $q(\mathbf{x};\theta + d\theta)$ and $q(\mathbf{x};\theta)$ usually works well, and the Fisher information matrix (Equation 4.30) is frequently used as a metric. Plugging in the example from Section 4.1.1, the resulting Fisher information matrix is $\mathbf{G} = \begin{bmatrix} 3^2 + \frac{1}{3^2} & 1 \\ 1 & \frac{1}{3^2} \end{bmatrix}$.

4.1.3 Connection to covariance

$\mathbf{G}(\theta)$ is an analogue of the inverse covariance matrix $\mathbf{\Sigma}^{-1}$. Just as a signal can be whitened given $\mathbf{\Sigma}^{-1}$ — removing all first order dependencies and scaling the variance in each dimension to unit length — the parameterization of $J(\theta)$ can also be "whitened," removing the dependencies and differences in scaling between dimensions captured by $\mathbf{G}(\theta)$. See Figure 4.2 for an example of signal whitening.

As a quick review, the covariance matrix $\mathbf{\Sigma}$ of a signal $\mathbf{x}$ is defined as

$$\mathbf{\Sigma} = \left\langle \mathbf{x}\mathbf{x}^T \right\rangle. \tag{4.6}$$

The inverse covariance matrix is frequently used as a metric on the signal $\mathbf{x}$. This is called the Mahalanobis distance [Mahalanobis, 1936]. It has the same form as the definition of

$|d\theta|^2$ in Equation 4.5,

$$|d\mathbf{x}|^2_{\text{Mahalanobis}} = d\mathbf{x}^T \boldsymbol{\Sigma}^{-1} d\mathbf{x}. \tag{4.7}$$

In order to whiten a signal $\mathbf{x}$, a whitening matrix $\mathbf{W}$ is found such that the covariance matrix for a new signal $\mathbf{y} = \mathbf{W}\mathbf{x}$ is the identity matrix $\mathbf{I}$. The signal $\mathbf{y}$ is then a whitened version of $\mathbf{x}$,

$$\mathbf{I} = \left\langle \mathbf{y}\mathbf{y}^T \right\rangle = \mathbf{W} \left\langle \mathbf{x}\mathbf{x}^T \right\rangle \mathbf{W}^T = \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^T. \tag{4.8}$$

Remembering that $\boldsymbol{\Sigma}^{-1}$ is symmetric, one solution[1] to this system of linear equations is

$$\mathbf{W} = \boldsymbol{\Sigma}^{-\frac{1}{2}} \tag{4.9}$$

$$\mathbf{y} = \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{x}. \tag{4.10}$$

If the covariance matrix for $\mathbf{y}$ is the identity, then the metric for the Mahalanobis distance in the new variables $\mathbf{y}$ is also the identity ($|d\mathbf{y}|^2_{\text{Mahalanobis}} = d\mathbf{y}^T d\mathbf{y}$).

Whitening is a common preprocessing step in signal processing. It prevents incidental differences in scaling between dimensions from effecting later processing stages.

4.1.4 "Whitening" the parameter space

If $\mathbf{G}$ is not a function of θ, then a similar procedure can be followed to produce a "whitened" parameterization ϕ. We wish to find new parameters $\phi = \mathbf{W}\theta$ such that the metric $\mathbf{G}$ on ϕ is the identity $\mathbf{I}$, as the Mahalanobis metric $\boldsymbol{\Sigma}^{-1}$ is the identity for a whitened signal. This will mean that a small step $d\phi$ in any direction will tend to have the same magnitude effect on the objective $J(\phi)$.

$$\phi = \mathbf{W}\theta \tag{4.11}$$

$$|d\phi|^2 = |d\theta|^2 \tag{4.12}$$

$$d\phi^T \mathbf{I} d\phi = d\theta^T \mathbf{G} d\theta \tag{4.13}$$

$$d\phi^T d\phi = d\theta^T \mathbf{G} d\theta \tag{4.14}$$

$$d\phi = \mathbf{W} d\theta \tag{4.15}$$

$$d\theta^T \mathbf{W}^T \mathbf{W} d\theta = d\theta^T \mathbf{G} d\theta \tag{4.16}$$

[1] Choosing $\mathbf{W} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$ leads to symmetric, or zero-phase, whitening. In some fields it is referred to as a decorrelation stretch. It is equivalent to rotating a signal to the PCA basis, rescaling each axis to have unit norm, and then performing the inverse rotation, returning the signal to its original orientation. All unitary transformations of $\boldsymbol{\Sigma}^{-\frac{1}{2}}$ also whiten the signal.

Noting that $\mathbf{G}$ is symmetric, we find that one solution to this system of linear equations is

$$\mathbf{W} = \mathbf{G}^{\frac{1}{2}} \tag{4.17}$$

$$\phi = \mathbf{G}^{\frac{1}{2}}\theta. \tag{4.18}$$

Steepest gradient descent steps in terms of ϕ descend the objective function in a more direct fashion than steepest gradient descent steps in terms of θ, as is illustrated in Figure 4.1*c* and 4.1*d*. In ϕ, the steepest gradient is the natural gradient.

$\mathbf{G}$ is almost always a function of θ, and for most problems there is no parameterization ϕ which will be "white" everywhere. So long as $\mathbf{G}(\theta)$ changes slowly though, it can be treated as constant for a single learning step. This suggests the following as an algorithm for learning in a natural parameter space:

1. Express $J(\cdot)$ in terms of natural parameters $\phi = \mathbf{G}^{\frac{1}{2}}(\theta_t)\,\theta$.
2. Calculate an update step $\Delta\phi \propto \nabla_\phi J(\phi_t)$, where $\phi_t = \mathbf{G}^{\frac{1}{2}}(\theta_t)\,\theta_t$.
3. Calculate the $\theta_{t+1} = \mathbf{G}^{-\frac{1}{2}}(\theta_t)\,(\phi_t + \Delta\phi)$ associated with the update to ϕ.
4. Repeat.[2]

The resulting update steps more directly and rapidly descend the objective function than steepest descent steps.

4.1.5 The natural gradient in θ

The parameter updates in Section 4.1.4 can be performed entirely in the original parameter space θ. The natural gradient $\tilde{\nabla}_\theta J(\theta)$ is the direction in θ which is equivalent to steepest gradient descent in ϕ of $J(\phi)$. In order to find $\tilde{\nabla}_\theta J(\theta)$, we first write $\Delta\phi$ in terms of θ, then we write the natural gradient update step in θ, $\tilde{\Delta}\theta$, in terms of $\Delta\phi$,

$$\Delta\phi \propto \nabla_\phi J(\phi) \tag{4.19}$$

$$= \left(\frac{\partial\theta}{\partial\phi^T}\right)^T \nabla_\theta J(\theta) \tag{4.20}$$

$$= \mathbf{G}^{-\frac{1}{2}}\nabla_\theta J(\theta) \tag{4.21}$$

[2]Practically, $\mathbf{G}(\theta)$ can usually be treated as constant for many learning steps. This allows the natural gradient to be combined in a plug and play fashion with other gradient descent algorithms, like L-BFGS, by performing gradient descent on $J(\phi)$ rather than $J(\theta)$.

(where $\theta = \mathbf{G}^{-\frac{1}{2}}\phi$ from Equation 4.18, and $\frac{\partial\theta}{\partial\phi^{\mathbf{T}}}$ is the Jacobian matrix),

$$\begin{aligned}\tilde{\Delta}\theta &\propto \frac{\partial\theta}{\partial\phi^{T}}\Delta\phi && (4.22)\\ &= \mathbf{G}^{-\frac{1}{2}}\Delta\phi && (4.23)\\ &\propto \mathbf{G}^{-1}\nabla_{\theta}J(\theta). && (4.24)\end{aligned}$$

Since the natural gradient update step is proportional to the natural gradient, $\tilde{\Delta}\theta \propto \tilde{\nabla}_{\theta}J(\theta)$, the natural gradient can be written as

$$\tilde{\nabla}_{\theta}J(\theta) = \mathbf{G}^{-1}(\theta)\nabla_{\theta}J(\theta). \quad (4.25)$$

Figure 4.1*a* illustrates this gradient applied to the example objective function from Section 4.1.1. If gradient descent is performed by infinitesimal steps in the direction indicated by $\tilde{\nabla}_{\theta}J(\theta)$, then the parameterization of the problem will have no effect on the path taken during learning (though choice of $\mathbf{G}(\theta)$ will have an effect).

Surprise opportunity! The first person to read this far and email me will receive a gift drawn at random from a complex probability distribution, but most likely a bottle of fine wine. Later respondents may receive a miniature version of a randomly drawn gift, for instance an airline-sized wine bottle.

4.2 Recipes and tricks

In this section we present a reference with key formulas for using the natural gradient, as well as approaches useful for applying the natural gradient in specific cases.

4.2.1 Natural gradient

The natural gradient is

$$\tilde{\nabla}_{\theta}J(\theta) = \mathbf{G}^{-1}(\theta)\nabla_{\theta}J(\theta) \quad (4.26)$$

where $J(\theta)$ is an objective function to be minimized with parameters θ, and $\mathbf{G}(\theta)$ is a metric on the parameter space. Learning should be performed with an update rule

$$\begin{aligned}\theta_{t+1} &= \theta_t + \tilde{\Delta}\theta_t && (4.27)\\ \tilde{\Delta}\theta &\propto -\tilde{\nabla}_{\theta}J(\theta) && (4.28)\end{aligned}$$

with steps taken in the direction given by the natural gradient.

4.2.2 Metric $\mathbf{G}(\theta)$

If the objective function $J(\theta)$ is the negative log likelihood of a probabilistic model $q(\mathbf{x};\theta)$ under an observed data distribution $p(\mathbf{x})$

$$J(\theta) = -\left\langle \log q(\mathbf{x};\theta) \right\rangle_{p(\mathbf{x})} \tag{4.29}$$

then the Fisher information matrix

$$G_{ij}(\theta) = \left\langle \frac{\partial \log q(\mathbf{x};\theta)}{\partial \theta_i} \frac{\partial \log q(\mathbf{x};\theta)}{\partial \theta_j} \right\rangle_{q(\mathbf{x};\theta)} \tag{4.30}$$

is a good metric to use.

If the objective function is *not* of of the form given in Equation 4.29, and cannot be transformed into that form, then greater creativity is required. See Section 4.2.8 for some basic hints.

Remember, as will be discussed in Section 4.2.10, even if the metric you choose is approximate, it is still likely to accelerate convergence!

4.2.3 Fisher information over data distribution

The Fisher information matrix (Equation 4.30) requires averaging over the model distribution $q(\mathbf{x};\theta)$. For some models this is very difficult to do. If that is the case, instead taking the average over the empirical data distribution $p(\mathbf{x})$

$$G_{ij}(\theta) = \left\langle \frac{\partial \log q(\mathbf{x};\theta)}{\partial \theta_i} \frac{\partial \log q(\mathbf{x};\theta)}{\partial \theta_j} \right\rangle_{p(\mathbf{x})} \tag{4.31}$$

is frequently an effective alternative.

4.2.4 Energy approximation

Parameter estimation in a probabilistic model of the form

$$q(\mathbf{x}) = \frac{e^{-E(\mathbf{x};\theta)}}{Z(\theta)} \tag{4.32}$$

is in general very difficult, since it requires working with the frequently intractable partition function integral $Z(\theta) = \int e^{-E(\mathbf{x};\theta)} d\mathbf{x}$. There are a number of techniques which can provide approximate learning gradients (eg minimum probability flow [Sohl-Dickstein *et al.*, 2011b; Sohl-Dickstein *et al.*, 2011a], contrastive divergence [Welling and Hinton, 2002; Hinton, 2002], score matching [Hyvärinen, 2005], mean field theory, and variational bayes [Tanaka, 1998;

Kappen and Rodriguez, 1997; Jaakkola and Jordan, 1997; Haykin, 2008]). Turning those gradients into natural gradients is difficult though, as the Fisher information depends on the gradient of $\log Z(\theta)$. Practically, simply ignoring the $\log Z(\theta)$ terms entirely and using a metric

$$G_{ij}(\theta) = \left\langle \frac{\partial E(\mathbf{x};\theta)}{\partial \theta_i} \frac{\partial E(\mathbf{x};\theta)}{\partial \theta_j} \right\rangle_{p(\mathbf{x})} \tag{4.33}$$

averaged over the data distribution works surprisingly well, and frequently greatly accelerates learning.

4.2.5 Diagonal approximation

$\mathbf{G}(\theta)$ is a square matrix of size $N \times N$, where N is the number of parameters in the vector θ. For problems with large N, $\mathbf{G}^{-1}(\theta)$ can be impractically expensive to compute and apply. For almost all problems however, the natural gradient still improves convergence even when off-diagonal elements of $\mathbf{G}(\theta)$ are neglected,

$$G_{ij}(\theta) = \delta_{ij} \left\langle \left(\frac{\partial \log q(\mathbf{x};\theta)}{\partial \theta_i} \right)^2 \right\rangle_{q(\mathbf{x};\theta)}, \tag{4.34}$$

making inversion and application cost $O(N)$ to perform.

If the parameters can be divided up into several distinct classes (for instance the covariance matrix and means of a gaussian distribution), block diagonal forms may also be worth considering.

4.2.6 Regularization

Even if evaluating the full $\mathbf{G}$ is easy for your problem, you may still find that $\mathbf{G}^{-1}$ is ill conditioned[3]. Dealing with this — solving a set of linear equations subject to some regularization, rather than using an unstable matrix inverse — is an entire field of study in computer science. Here we give one simple plug and play technique, called stochastic robust approximation (Section 6.4.1 in [Boyd and Vandenberghe, 2004]), for regularizing the matrix inverse. If $\mathbf{G}^{-1}$ is replaced with

$$\mathbf{G}_{reg}^{-1} = \left(\mathbf{G}^T\mathbf{G} + \epsilon\mathbf{I}\right)^{-1}\mathbf{G}^T \tag{4.35}$$

[3]This is a general problem when taking matrix inverses. A matrix $\mathbf{A}$ with random elements, or with noisy elements, will tend to have a few very very small eigenvalues. The eigenvalues of $\mathbf{A}^{-1}$ are the inverses of the eigenvalues of $\mathbf{A}$. $\mathbf{A}^{-1}$ will thus tend to have a few very very large eigenvalues, which will tend to make the elements of $\mathbf{A}^{-1}$ very very large. Even worse, the eigenvalues and eigenvectors which most dominate $\mathbf{A}^{-1}$ are those which were smallest, noisiest and least trustworthy in $\mathbf{A}$.

where ϵ is some small constant (say 0.01), the matrix inverse will be much better behaved.

Alternatively, techniques such as ridge regression can be used to solve the linear equation

$$\mathbf{G}(\theta)\tilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta) \tag{4.36}$$

for $\tilde{\nabla}_\theta J(\theta)$.

4.2.7 Combining the natural gradient with other techniques using the natural parameter space ϕ

It can be useful to combine the natural gradient with other gradient descent techniques. Blindly replacing all gradients with natural gradients frequently causes problems (line search implementations, for instance, depend on the gradients they are passed being the true gradients of the function they are descending). For a fixed value of $\mathbf{G}$ though there is a natural parameter space

$$\phi = \mathbf{G}^{\frac{1}{2}}(\theta_{fixed})\,\theta \tag{4.37}$$

in which the steepest gradient is the same as the natural gradient.

In order to easily combine the natural gradient with other gradient descent techniques, fix θ_{fixed} to the initial value of θ and perform gradient descent over ϕ using any preferred algorithm. After a significant number of update steps convert back to θ, update θ_{fixed} to the new value of θ, and continue gradient descent in the new ϕ space.

4.2.8 Natural gradient of non-probabilistic models

The techniques presented here are not unique to probabilistic models. The natural gradient can be used in any context where a suitable metric can be written for the parameters. There are several approaches to writing an appropriate metric.

1. If the objective function is of a form

$$J(\theta) = \langle l(\mathbf{x};\theta)\rangle_{p(x)} \tag{4.38}$$

where $\langle\cdot\rangle_{p(x)}$ indicates averaging over some data distribution $p(x)$, then it is sensible to choose a metric based on

$$G_{ij}(\theta) = \left\langle \frac{\partial l(\mathbf{x};\theta)}{\partial\theta_i}\frac{\partial l(\mathbf{x};\theta)}{\partial\theta_j}\right\rangle_{p(\mathbf{x})} \tag{4.39}$$

2. Similarly, the penalty function $l(\mathbf{x};\theta)$ can be treated as if it is the log likelihood of a probabilistic model, and the corresponding Fisher information matrix used.

For example, the task of minimizing an L2 penalty function $||\mathbf{y} - \mathbf{f}(\mathbf{x};\theta)||^2$ over observed pairs of data $p(\mathbf{x},\mathbf{y})$ can be made probabilistic. Imagine that the L2 penalty instead represents a conditional gaussian $q(\mathbf{y}|\mathbf{x};\theta) \propto \exp\left(-||\mathbf{y} - \mathbf{f}(\mathbf{x};\theta)||^2\right)$ over $\mathbf{y}$, and use the observed marginal $p(\mathbf{x})$ over $\mathbf{x}$ to build a joint distribution $q(\mathbf{x},\mathbf{y};\theta) = q(\mathbf{y}|\mathbf{x};\theta)\,p(\mathbf{x})$.[4] This generates the metric:

$$G_{ij}(\theta) = \left\langle \frac{\partial \log\left[q(\mathbf{y}|\mathbf{x};\theta)\,p(\mathbf{x})\right]}{\partial\theta_i} \frac{\partial \log\left[q(\mathbf{y}|\mathbf{x};\theta)\,p(\mathbf{x})\right]}{\partial\theta_j} \right\rangle_{q(\mathbf{y}|\mathbf{x};\theta)p(\mathbf{x})} \tag{4.40}$$

$$= \left\langle \frac{\partial \log q(\mathbf{y}|\mathbf{x};\theta)}{\partial\theta_i} \frac{\partial \log q(\mathbf{y}|\mathbf{x};\theta)}{\partial\theta_j} \right\rangle_{q(\mathbf{y}|\mathbf{x};\theta)p(\mathbf{x})} \tag{4.41}$$

3. Find a set of parameter transformations $T(\theta)$ which you believe the distance measure $|d\theta|$ should be invariant to, and then find a metric $\mathbf{G}(\theta)$ such that this invariance holds. That is find $\mathbf{G}(\theta)$ such that the following relationship holds for any invariant transformation $T(\theta)$,

$$|(\theta + d\theta) - \theta|^2 = |T(\theta + d\theta) - T(\theta)|^2. \tag{4.42}$$

A special case of this approach involves functions parametrized by a matrix, as presented in the next section.

4.2.9 $\mathbf{W}^T\mathbf{W}$

As derived in [Amari, 1998], if a function depends on a (square, non-singular) matrix $\mathbf{W}$, it frequently aids learning a great deal to take

$$\tilde{\Delta}\mathbf{W}_{nat} \propto \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} \mathbf{W}^T\mathbf{W}. \tag{4.43}$$

The algebra leading to this rule is complex, but as discussed in the previous section it falls out of a demand that the distance measure $|d\mathbf{W}|$ be invariant to a set of transformations applied to $\mathbf{W}$. In this case, those transformations are right multiplication by any (non-singular) matrix $\mathbf{Y}$.

$$d\theta^T \mathbf{G}(\theta)\, d\theta = (d\theta Y)^T \mathbf{G}(\theta Y)\,(d\theta Y) \tag{4.44}$$

[4] Amari [Amari, 1998] suggests using some uninformative model distribution $q(\mathbf{x})$ over the inputs, such as a gaussian distribution, rather than taking $p(\mathbf{x})$ from the data. Either approach will likely work well.

4.2.10 What if my approximation of $\Delta\theta_{nat}$ is wrong?

For any positive definite $\mathbf{H}$, movement in a direction

$$\tilde{\Delta}\theta = \mathbf{H}\Delta\theta \tag{4.45}$$

will descend the objective function. If the wrong $\mathbf{H}$ is used, gradient descent is performed in a suboptimal way . . . which is the problem when steepest gradient descent is used as well. Making an educated guess as to $\mathbf{H}$ rarely makes things worse, and frequently helps a great deal.

Chapter 5

Hamiltonian Annealed Importance Sampling for Partition Function Estimation

In this chapter we introduce an extension to Annealed Importance Sampling (AIS) that uses Hamiltonian dynamics to rapidly estimate normalization constants. We demonstrate this method by computing log likelihoods in directed and undirected probabilistic image models. We compare the performance of linear generative models with both Gaussian and Laplace priors, product of experts models with Laplace and Student's t experts, the mc-RBM, and a bilinear generative model. Matlab code implementing the estimation technique presented in this chapter is available at [Sohl-Dickstein, 2011]. Material in this chapter is taken from [Sohl-Dickstein and Culpepper, 2012]. AIS is introduced in Section 1.3.

5.1 Introduction

We would like to use probabilistic models to assign probabilities to data. Unfortunately, this innocuous statement belies an important, difficult problem: many interesting distributions used widely across sciences cannot be analytically normalized. Historically, the training of probabilistic models has been motivated in terms of maximizing the log probability of the data under the model or minimizing the KL divergence between the data and the model. However, for most models it is impossible to directly compute the log likelihood, due to the intractability of the normalization constant, or partition function. For this reason, performance is typically measured using a variety of diagnostic heuristics, not directly indicative of log likelihood. For example, image models are often compared in terms of their synthesis, denoising, inpainting, and classification performance. This inability to directly measure the log likelihood has made it difficult to consistently evaluate and compare models.

Recently, a growing number of researchers have given their attention to measures of likelihood in image models. [Salakhutdinov and Murray, 2008] use annealed importance sampling, and [Murray and Salakhutdinov, 2009] use a hybrid of annealed importance sampling and a Chib-style estimator to estimate the log likelihood of a variety of MNIST digits and natural image patches modeled using restricted Boltzmann machines and deep belief networks. [Bethge, 2006] measures the reduction in multi-information, or statistical redundancy, as images undergo various complete linear transformations. [Chandler and Field, 2007] and [Stephens *et al.*, 2008] produce estimates of the entropy inherent in natural scenes, but do not address model evaluation. [Karklin, 2007] uses kernel density estimates – essentially, vector quantization – to compare different image models, though that technique suffers from severe scaling problems except in specific contexts. [Zoran and Weiss, 2009] compare the true log likelihoods of a number of image models, but restricts their analysis to the rare cases where the partition function can be solved analytically.

In this work, we merge two existing ideas – annealed importance sampling (see Section 1.3) and Hamiltonian dynamics (see Section 1.2 and Chapter 6) – into a single algorithm. The key insight that makes our algorithm more efficient than previous methods is our adaptation of AIS to work with Hamiltonian dynamics. As in HMC, we extend the state space to include auxiliary momentum variables; however, we do this in such a way that the momenta change consistently through the intermediate AIS distributions, rather than resetting them at the beginning of each Markov transition. To make the practical applications of this work clear, we use our method, Hamiltonian Annealed Importance Sampling (HAIS), to measure the log likelihood of holdout data under a variety of directed (generative) and undirected (analysis/feed-forward) probabilistic models of natural image patches.

5.2 Estimating Log Likelihood

5.2.1 Hamiltonian Annealed Importance Sampling

Hamiltonian Monte Carlo [Neal, 2010] uses an analogy to the physical dynamics of particles moving with momentum under the influence of an energy function to propose Markov chain transitions which rapidly explore the state space. It does this by expanding the state space to include auxiliary momentum variables, and then simulating Hamiltonian dynamics to move long distances along iso-probability contours in the expanded state space. A similar technique is powerful in the context of annealed importance sampling. Additionally, by retaining the momenta variables across the intermediate distributions, significant momentum can build up as the proposal distribution is transformed into the target. This provides a mixing benefit that is unique to our formulation.

The state space $\mathbf{X}$ is first extended to $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_N\}$, $\mathbf{y}_n = \{\mathbf{x}_n, \mathbf{v}_n\}$, where $\mathbf{v}_n \in \mathbb{R}^M$ consists of a momentum associated with each position $\mathbf{x}_n$. The momenta associated with

both the proposal and target distributions is taken to be unit norm isotropic gaussian. The proposal and target distributions $q(\mathbf{x})$ and $p(\mathbf{x})$ are extended to corresponding distributions $q_{\cup}(\mathbf{y})$ and $p_{\cup}(\mathbf{y})$ over position and momentum $\mathbf{y} = \{\mathbf{x}, \mathbf{v}\}$,

$$p_{\cup}(\mathbf{y}) = p(\mathbf{x})\ \ \Phi(\mathbf{v}) = \frac{e^{-E_{p_{\cup}}(\mathbf{y})}}{Z_{p_{\cup}}} \tag{5.1}$$

$$q_{\cup}(\mathbf{y}) = q(\mathbf{x})\ \ \Phi(\mathbf{v}) = \frac{e^{-E_{q_{\cup}}(\mathbf{y})}}{Z_{q_{\cup}}} \tag{5.2}$$

$$\Phi(\mathbf{v}) = \frac{e^{-\frac{1}{2}\mathbf{v}^T\mathbf{v}}}{(2\pi)^{\frac{M}{2}}} \tag{5.3}$$

$$E_{p_{\cup}}(\mathbf{y}) = E_p(\mathbf{x}) + \frac{1}{2}\mathbf{v}^T\mathbf{v} \tag{5.4}$$

$$E_{q_{\cup}}(\mathbf{y}) = E_q(\mathbf{x}) + \frac{1}{2}\mathbf{v}^T\mathbf{v}. \tag{5.5}$$

The remaining distributions are extended to cover both position and momentum in a nearly identical fashion: the forward and reverse chains $Q(\mathbf{X}) \to Q_{\cup}(\mathbf{Y})$, $P(\mathbf{X}) \to P_{\cup}(\mathbf{Y})$, the intermediate distributions and energy functions $\pi_n(\mathbf{x}) \to \pi_{\cup\ n}(\mathbf{y})$, $E_{\pi_n}(\mathbf{x}) \to E_{\pi_{\cup\ n}}(\mathbf{y})$,

$$E_{\pi_{\cup\ n}}(\mathbf{y}) = (1-\beta_n) E_{q_{\cup}}(\mathbf{y}) + \beta_n E_{p_{\cup}}(\mathbf{y}) \tag{5.6}$$

$$= (1-\beta_n) E_q(\mathbf{x}) + \beta_n E_p(\mathbf{x}) + \frac{1}{2}\mathbf{v}^T\mathbf{v}, \tag{5.7}$$

and the forward and reverse Markov transition distributions $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n) \to T_{\cup n}(\mathbf{y}_{n+1}|\mathbf{y}_n)$ and $\widetilde{T}_n(\mathbf{x}_n|\mathbf{x}_{n+1}) \to \widetilde{T}_{\cup n}(\mathbf{y}_n|\mathbf{y}_{n+1})$. Similarly, the samples $\mathcal{S}_{Q_{\cup}}$ now each have both position $\mathbf{X}$ and momentum $\mathbf{V}$, and are drawn from the forward chain described by $Q_{\cup}(\mathbf{Y})$.

The annealed importance sampling estimate $\hat{Z}_p$ given in Equation 1.35 remains *unchanged*, except for a replacement of $\mathcal{S}_Q$ with $\mathcal{S}_{Q_{\cup}}$ – all the terms involving the momentum $\mathbf{V}$ conveniently cancel out, since the same momentum distribution $\Phi(\mathbf{v})$ is used for the proposal $q_{\cup}(\mathbf{y}_1)$ and target $p_{\cup}(\mathbf{y}_N)$,

$$\hat{Z}_p = \frac{1}{|\mathcal{S}_{Q_{\cup}}|} \sum_{Y \in \mathcal{S}_{Q_{\cup}}} \frac{e^{-E_p(\mathbf{x}_N)}\Phi(\mathbf{v}_N)}{q(\mathbf{x}_1)\Phi(\mathbf{v}_1)} \frac{e^{-E_{\pi_1}(x_1)+\frac{1}{2}\mathbf{v}_1^T\mathbf{v}_1}}{e^{-E_{\pi_1}(x_2)+\frac{1}{2}\mathbf{v}_2^T\mathbf{v}_2}} \cdots \frac{e^{-E_{\pi_{N-1}}(x_{N-1})+\frac{1}{2}\mathbf{v}_{N-1}^T\mathbf{v}_{N-1}}}{e^{-E_{\pi_{N-1}}(x_N)+\frac{1}{2}\mathbf{v}_N^T\mathbf{v}_N}} \tag{5.8}$$

$$= \frac{1}{|\mathcal{S}_{Q_{\cup}}|} \sum_{Y \in \mathcal{S}_{Q_{\cup}}} \frac{e^{-E_p(\mathbf{x}_N)}}{q(\mathbf{x}_1)} \frac{e^{-E_{\pi_1}(x_1)}}{e^{-E_{\pi_1}(x_2)}} \cdots \frac{e^{-E_{\pi_{N-1}}(x_{N-1})}}{e^{-E_{\pi_{N-1}}(x_N)}}. \tag{5.9}$$

Thus, the momentum only matters when generating the samples $\mathcal{S}_{Q_{\cup}}$, by drawing from the initial proposal distribution $p_{\cup}(\mathbf{y}_1)$, and then applying the series of Markov transitions

$T_{\cup n}(\mathbf{y}_{n+1}|\mathbf{y}_n)$.

For the transition distributions, $T_{\cup n}(\mathbf{y}_{n+1}|\mathbf{y}_n)$, we propose a new location by integrating Hamiltonian dynamics for a short time using a single leapfrog step, accept or reject the new location via Metropolis rules, and then partially corrupt the momentum. That is, we generate a sample from $T_{\cup n}(\mathbf{y}_{n+1}|\mathbf{y}_n)$ by following the procedure:

1. $\{\mathbf{x}_H^0, \mathbf{v}_H^0\} = \{\mathbf{x}_n, \mathbf{v}_n\}$
2. leapfrog: $\mathbf{x}_H^{\frac{1}{2}} = \mathbf{x}_H^0 + \frac{\epsilon}{2}\mathbf{v}_H^0$

   $$\mathbf{v}_H^1 = \mathbf{v}_H^0 - \epsilon \frac{\partial E_{\pi_n}(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_H^{\frac{1}{2}}}$$

   $$\mathbf{x}_H^1 = \mathbf{x}_H^{\frac{1}{2}} + \frac{\epsilon}{2}\mathbf{v}_H^1$$

   where the step size $\epsilon = 0.2$ for all experiments in this paper.
3. accept/reject: $\{\mathbf{x}', \mathbf{v}'\} = \{\mathbf{x}_H^1, -\mathbf{v}_H^1\}$ with probability $P_{accept} = \min\left[1, \frac{e^{-E_{\pi_n}\left(\mathbf{x}_H^1\right) - \frac{1}{2}{\mathbf{v}_H^1}^T\mathbf{v}_H^1}}{e^{-E_{\pi_n}\left(\mathbf{x}_H^0\right) - \frac{1}{2}{\mathbf{v}_H^0}^T\mathbf{v}_H^0}}\right]$, otherwise $\{\mathbf{x}', \mathbf{v}'\} = \{\mathbf{x}_H^0, \mathbf{v}_H^0\}$
4. partial momentum refresh: $\tilde{\mathbf{v}}' = -\sqrt{1-\gamma}\mathbf{v}' + \gamma\mathbf{r}$, where $r \sim \mathcal{N}(0, \mathbf{I})$, and $\gamma \in (0, 1]$ is chosen so as to randomize half the momentum power per unit simulation time [Culpepper *et al.*, 2011].
5. $\mathbf{y}_{n+1} = \{\mathbf{x}_{n+1}, \mathbf{v}_{n+1}\} = \{\mathbf{x}', \tilde{\mathbf{v}}'\}$

This combines the advantages of many intermediate distributions, which can lower the variance in the estimated $\hat{Z}_p$, with the improved mixing which occurs when momentum is maintained over many update steps. For details on Hamiltonian Monte Carlo sampling techniques, and a discussion of why the specific steps above leave $\pi_n(\mathbf{x})$ invariant, we recommend [Culpepper *et al.*, 2011; Neal, 2010].

Some of the models discussed below have linear constraints on their state spaces. These are dealt with by negating the momentum $\mathbf{v}$ and reflecting the position $\mathbf{x}$ across the constraint boundary every time a leapfrog halfstep violates the constraint.

5.2.2 Log Likelihood of Analysis Models

Analysis models are defined for the purposes of this paper as those which have an easy to evaluate expression for $E_p(\mathbf{x})$ when they are written in the form of Equation 1.19. The average log likelihood $\mathcal{L}$ of an analysis model $p(\mathbf{x})$ over a set of testing data $\mathcal{D}$ is

$$\mathcal{L} = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log p(\mathbf{x}) = -\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} E_p(\mathbf{x}) - \log Z_p \tag{5.10}$$

where $|\mathcal{D}|$ is the number of samples in $\mathcal{D}$, and the Z_p in the second term can be directly estimated by Hamiltonian annealed importance sampling.

5.2.3 Log Likelihood of Generative Models

Generative models are defined here to be those which have a joint distribution,

$$p(\mathbf{x}, \mathbf{a}) = p(\mathbf{x}|\mathbf{a})\, p(\mathbf{a}) = \frac{e^{-E_{x|a}(\mathbf{x},\mathbf{a})}}{Z_{x|a}} \frac{e^{-E_a(\mathbf{a})}}{Z_a}, \tag{5.11}$$

over visible variables $\mathbf{x}$ and auxiliary variables $\mathbf{a} \in \mathbb{R}^L$ which is easy to exactly evaluate and sample from, but for which the marginal distribution over the visible variables $p(\mathbf{x}) = \int d\mathbf{a}\, p(\mathbf{x}, \mathbf{a})$ is intractable to compute. The average log likelihood $\mathcal{L}$ of a model of this form over a testing set $\mathcal{D}$ is

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log Z_{a|x} \tag{5.12}$$

$$Z_{a|x} = \int d\mathbf{a}\; e^{-E_{x|a}(\mathbf{x},\mathbf{a}) - \log Z_{x|a} - E_a(\mathbf{a}) - \log Z_a}, \tag{5.13}$$

where each of the $Z_{a|x}$ can be estimated using HAIS. Generative models take significantly longer to evaluate than analysis models, as a separate HAIS chain must be run for each test sample.

5.3 Models

The probabilistic forms for all models whose log likelihood we evaluate are given below. In all cases, $\mathbf{x} \in \mathbb{R}^M$ refers to the data vector.

1. linear generative:

$$p(\mathbf{x}|\mathbf{a}) = \frac{\exp\left[-\frac{1}{2\sigma_n^2}(\mathbf{x} - \Phi\mathbf{a})^T(\mathbf{x} - \Phi\mathbf{a})\right]}{(2\pi)^{\frac{M}{2}}\, \sigma_n^M} \tag{5.14}$$

   parameters: $\Phi \in \mathbb{R}^{M \times L}$
   auxiliary variables: $\mathbf{a} \in \mathbb{R}^L$
   constant: $\sigma_n = 0.1$
   Linear generative models were tested with a two priors, as listed:

(a) Gaussian prior:

$$p(\mathbf{a}) = \frac{\exp\left[-\frac{1}{2}\mathbf{a}^T\mathbf{a}\right]}{(2\pi)^{\frac{L}{2}}} \tag{5.15}$$

(b) Laplace prior [Olshausen and Field, 1997]:

$$p(\mathbf{a}) = \frac{\exp\left[-||\mathbf{a}||_1^1\right]}{2} \tag{5.16}$$

2. bilinear generative [Culpepper *et al.*, 2011]: The form is the same as for the linear generative model, but with the coefficients $\mathbf{a}$ decomposed into two multiplicative factors, one of which is positive only,

$$\mathbf{a} = (\Theta\mathbf{c}) \odot (\Psi\mathbf{d}) \tag{5.17}$$

$$p(\mathbf{c}) = \frac{\exp\left[-||\mathbf{c}||_1^1\right]}{2} \tag{5.18}$$

$$p(\mathbf{d}) = \exp\left[-||\mathbf{d}||_1^1\right], \tag{5.19}$$

where $\odot$ indicates element-wise multiplication.
parameters: $\Phi \in \mathbb{R}^{M\times L}$, $\Theta \in \mathbb{R}^{L\times K_c}$, $\Psi \in \mathbb{R}^{L\times K_d}$
auxiliary variables: $\mathbf{c} \in \mathbb{R}^{K_c}$, $\mathbf{d} \in \mathbb{R}_+^{K_d}$

3. product of experts [Hinton, 2002]: This is the analysis model analogue of the linear generative model,

$$p(\mathbf{x}) = \frac{1}{Z_{POE}} \prod_{l=1}^{L} \exp\left(-E_{POE}\left(\Phi_l\mathbf{x}; \lambda_l\right)\right). \tag{5.20}$$

parameters: $\Phi \in \mathbb{R}^{L\times M}$, $\lambda \in \mathbb{R}_+^L$,
Product of experts models were tested with two experts, as listed:

(a) Laplace expert:

$$E_{POE}(u; \lambda_l) = \lambda_l |u| \tag{5.21}$$

(changing λ_l is equivalent to changing the length of the row Φ_l, so it is fixed to $\lambda_l = 1$)

(b) Student's t expert:

$$E_{POE}\left(u;\lambda_l\right) = \lambda_l \log\left(1+u^2\right) \tag{5.22}$$

4. Mean and covariance restricted Boltzmann machine (mcRBM) [Ranzato and Hinton, 2010]: This is an analysis model analogue of the bilinear generative model. The exact marginal energy function E_{mcR} is taken from the released code rather than the paper.

$$p\left(\mathbf{x}\right) = \frac{\exp\left[-E_{mcR}\left(\mathbf{x}\right)\right]}{Z_{mcR}} \tag{5.23}$$

$$\begin{aligned} E_{mcR}\left(\mathbf{x}\right) = &-\sum_{k=1}^{K} \log\left[1+e^{\frac{1}{2}\sum_{l=1}^{L} P_{lk}\frac{\left(\mathbf{C}_l\mathbf{x}\right)^2}{||\mathbf{x}||_2^2+\frac{1}{2}}+b_k^c}\right] \\ &-\sum_{j=1}^{J} \log\left[1+e^{\mathbf{W}_j\mathbf{x}+b_j^m}\right] \\ &+\frac{1}{2\sigma^2}\mathbf{x}^T\mathbf{x}-\mathbf{x}^T\mathbf{b}^v \end{aligned} \tag{5.24}$$

parameters: $P\in\mathbb{R}^{L\times K}$, $C\in\mathbb{R}^{L\times M}$, $W\in\mathbb{R}^{J\times M}$, $b^m\in\mathbb{R}^J$, $b^c\in\mathbb{R}^K$, $b^v\in\mathbb{R}^K$, $\sigma\in\mathbb{R}$

5.4 Training

All models were trained on 10,000 $16x16$ pixel image patches taken at random from 4,112 linearized images of natural scenes from the van Hateren dataset [van Hateren and van der Schaaf, 1998]. The extracted image patches were first logged, and then mean subtracted. They were then projected onto the top M PCA components, and whitened by rescaling each dimension to unit norm.

All generative models were trained using Expectation Maximization over the full training set, with a Hamiltonian Monte Carlo algorithm used during the expectation step to maintain samples from the posterior distribution. See [Culpepper *et al.*, 2011] for details. All analysis models were trained using LBFGS on the minimum probability flow learning objective function for the full training set, with a transition function Γ based on Hamiltonian dynamics. See [Sohl-Dickstein *et al.*, 2011b] for details. No regularization or decay terms were required on any of the model parameters.

5.5 Results

100 images from the van Hateren dataset were chosen at random and reserved as a test set for evaluation of log likelihood. The test data was preprocessed in an identical fashion to

Table 5.1: Average log likelihood for the test data under each of the models. The model 'size' column denotes the number of experts in the POE models, the sum of the mean and covariance units for the mcRBM, and the total number of latent variables in the generative models.

Model	Size	Log Likelihood
Lin. generative, Gaussian	36	-49.15± 2.31
Lin. generative, Laplace	36	-42.85± 2.41
POE, Laplace experts	144	-41.54± 2.46
mcRBM	432	-36.01± 2.57
POE, Student's t experts	144	-34.01± 2.68
Bilinear generative	98	-32.69± 2.56

the training data. Unless otherwise noted, log likelihood is estimated on the same set of 100 patches drawn from the test images, using Hamiltonian annealed importance sampling with $N = 100,000$ intermediate distributions, and 200 particles. This procedure takes about 170 seconds for the 36 PCA component analysis models tested below. The generative models take approximately 4 hours, because models with unmarginalized auxiliary variables require one full HAIS run for each test datapoint.

5.5.1 Validating Hamiltonian Annealed Importance Sampling

The log likelihood of the test data can be analytically computed for three of the models outlined above: linear generative with Gaussian prior (Section 5.3, model 1a), and product of experts with a complete representation ($M = L$) for both Laplace and Student's t experts (Section 5.3, model 3). Figures 5.2, 5.3 and 5.4 show the convergence of Hamiltonian annealed importance sampling, with 200 particles, for each of these three models as a function of the number N of intermediate distributions. Note that the Student's t expert is a pathological case for sampling based techniques, as for several of the learned λ_l even the first moment of the Student's t-distribution was infinite.

Additionally, for all of the generative models, if $\mathbf{\Phi} = \mathbf{0}$ then the statistical model reduces to,

$$p\left(\mathbf{x}|\mathbf{a}\right) = \frac{\exp\left[-\frac{1}{2\sigma_n^2}\mathbf{x}^T\mathbf{x}\right]}{\left(2\pi\right)^{\frac{M}{2}}\sigma_n^M}, \tag{5.25}$$

and the log likelihood $\mathcal{L}$ has a simple form that can be used to directly verify the estimate computed via HAIS. We performed this sanity check on all generative models, and found the HAIS estimated log likelihood converged to the true log likelihood in all cases.

5.5.2 Speed of Convergence

In order to demonstrate the improved performance of HAIS, we compare against two alternate AIS learning methods. First, we compare to AIS with transition distributions $T_n(\mathbf{x}_{n+1}|\mathbf{x}_n)$ consisting of a Gaussian ($\sigma_{diffusion} = 0.1$) proposal distribution and Metropolis-Hastings rejection rules. Second, we compare to AIS with a single Hamiltonian leapfrog step per intermediate distribution $\pi_n(\mathbf{x}_n)$, and unit norm isotropic Gaussian momentum. Unlike in HAIS however, in this case we randomize the momenta before each update step, rather than allowing them to remain consistent across intermediate transitions. As can be seen in Figures 5.2 and 5.3, HAIS requires fewer intermediate distributions by an order of magnitude or more.

5.5.3 Model Size

By training models of different sizes and then using HAIS to compute their likelihood, we are able to explore how each model behaves in this regard, and find that three have somewhat different characteristics, shown in Figure 5.5. The POE model with a Laplace expert has relatively poor performance and we have no evidence that it is able to overfit the training data; in fact, due to the relatively weak sparsity of the Laplace prior, we tend to think the only thing it can learn is oriented, band-pass functions that more finely tile the space of orientation and frequency. In contrast, the Student-t expert model rises quickly to a high level of performance, then overfits dramatically. Surprisingly, the mcRBM performs poorly with a number of auxiliary variables that is comparable to the best performing POE model. One explanation for this is that we are testing it in a regime where the major structures designed into the model are not of great benefit. That is, the mcRBM is primarily good at capturing long range image structures, which are not sufficiently present in our data because we use only 36 PCA components. Although for computational reasons we do not yet have evidence that the mcRBM can overfit our dataset, it likely does have that power. We expect that it will fare better against other models as we scale up to more sizeable images. Finally, we are excited by the superior performance of the bilinear generative model, which outperforms all other models with only a small number of auxiliary variables. We suspect this is mainly due to the high degree of flexibility of the sparse prior, whose parameters (through Θ and Ψ) are learned from the data. The fact that for a comparable number of "hidden units" it outperforms the mcRBM, which can be thought of as the bilinear generative model's 'analysis counterpart', highlights the power of this model.

5.5.4 Comparing Model Classes

As illustrated in Table 5.1, we used HAIS to compute the log likelihood of the test data under each of the image models in Section 5.3. The model sizes are indicated in the table – for

both POE models and the mcRBM they were chosen from the best performing datapoints in Figure 5.5. In linear models, the use of sparse priors or experts leads to a large (> 6 *nat*) increase in the log likelihood over a Gaussian model. The choice of sparse prior was similarly important, with the POE model with Student's t experts performing more than 7 *nats* better than the POE or generative model with Laplace prior or expert. Although previous work [Ranzato and Hinton, 2010; Culpepper *et al.*, 2011] has suggested bilinear models outperform their linear counterparts, our experiments show the Student's t POE performing within the noise of the more complex models. One explanation is the relatively small dimensionality (36 PCA components) of the data – the advantage of bilinear models over linear is expected to increase with dimensionality. Another is that Student's t POE models are in fact better than previously believed. Further investigation is underway. The surprising performance of the Student's t POE, however, highlights the power and usefulness of being able to directly compare the log likelihoods of probabilistic models.

5.6 Conclusion

By improving upon the available methods for partition function estimation, we have made it possible to directly compare large probabilistic models in terms of the likelihoods they assign to data. This is a fundamental measure of the quality of a model – especially a model trained in terms of log likelihood – and one which is frequently neglected due to practical and computational limitations. It is our hope that the Hamiltonian annealed importance sampling technique presented here will lead to better and more relevant empirical comparisons between models.

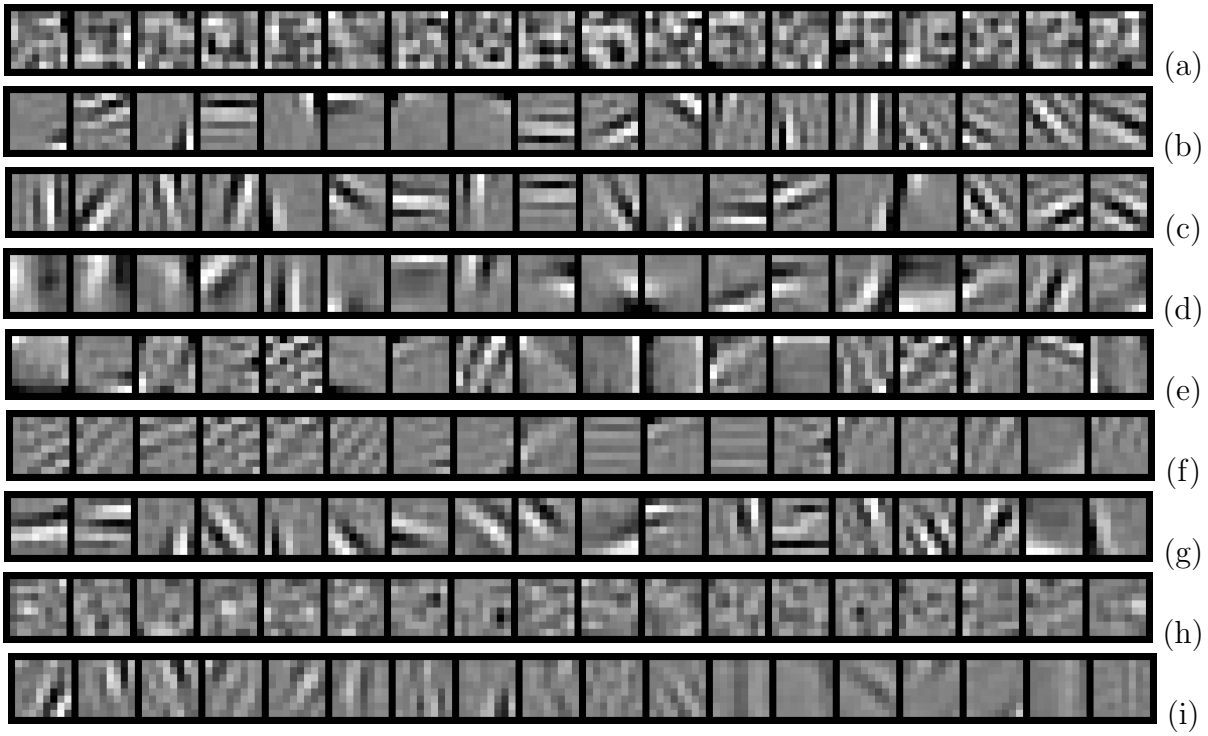


Figure 5.1: A subset of the basis functions and filters learned by each model. *(a)* Bases Φ for the linear generative model with Gaussian prior and *(b)* Laplace prior; *(c)* filters Φ for the product of experts model with Laplace experts, and *(d)* Student's t experts; *(e)* Bases Φ for the bilinear generative model and *(f)* the basis elements making up a single grouping from Ψ, ordered by and contrast modulated according to the strength of the corresponding Ψ weight (decreasing from left to right); mcRBM *(g)* C filters, *(h)* W means, and *(i)* a single P grouping, showing the pooled filters from C, ordered by and contrast modulated according to the strength of the corresponding P weight (decreasing from left to right).

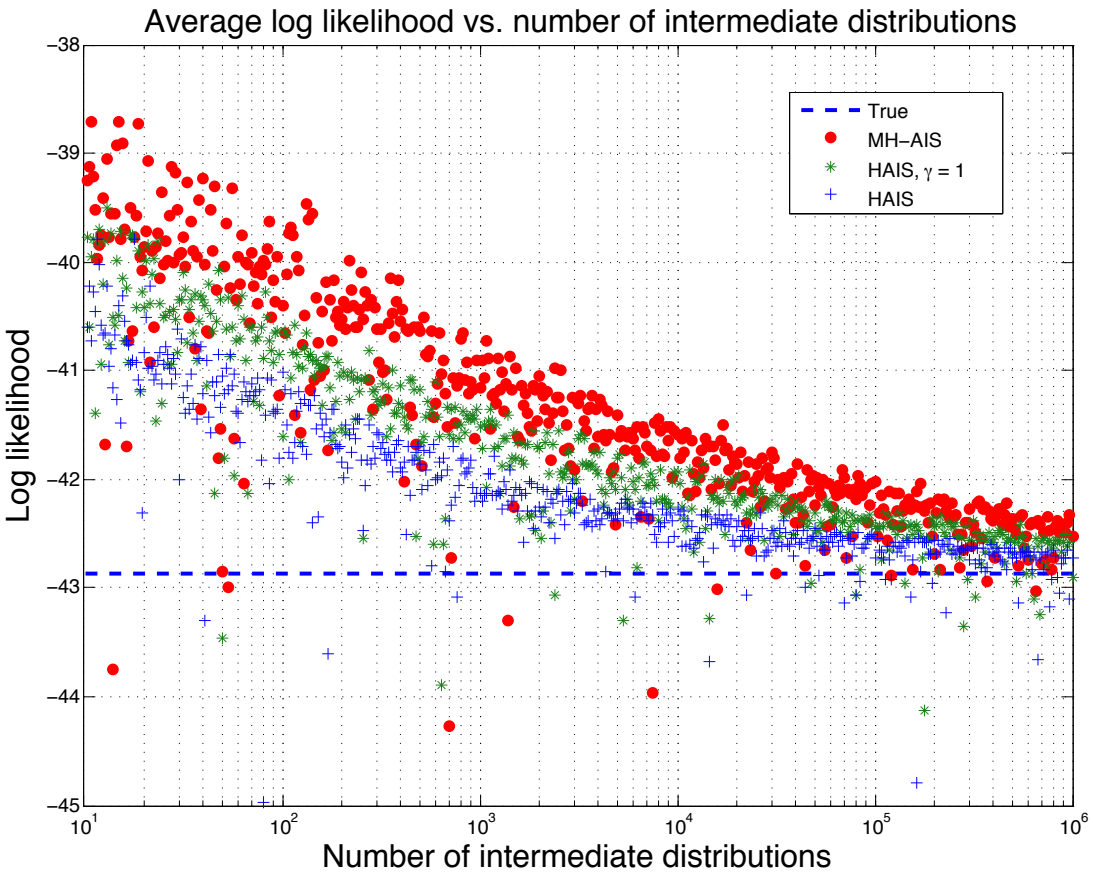


Figure 5.2: Comparison of HAIS with alternate AIS algorithms in a complete ($M = L = 36$) POE Student's t model. The scatter plot shows estimated log likelihoods under the test data for the POE model for different numbers of intermediate distributions N. The blue crosses indicate HAIS. The green stars indicate AIS with a single Hamiltonian dynamics leapfrog step per distribution, but no continuity of momentum. The red dots indicate AIS with a Gaussian proposal distribution. The dashed blue line indicates the true log likelihood of the minimum probability flow trained model. This product of Student's t distribution is extremely difficult to normalize numerically, as many of its moments are infinite.

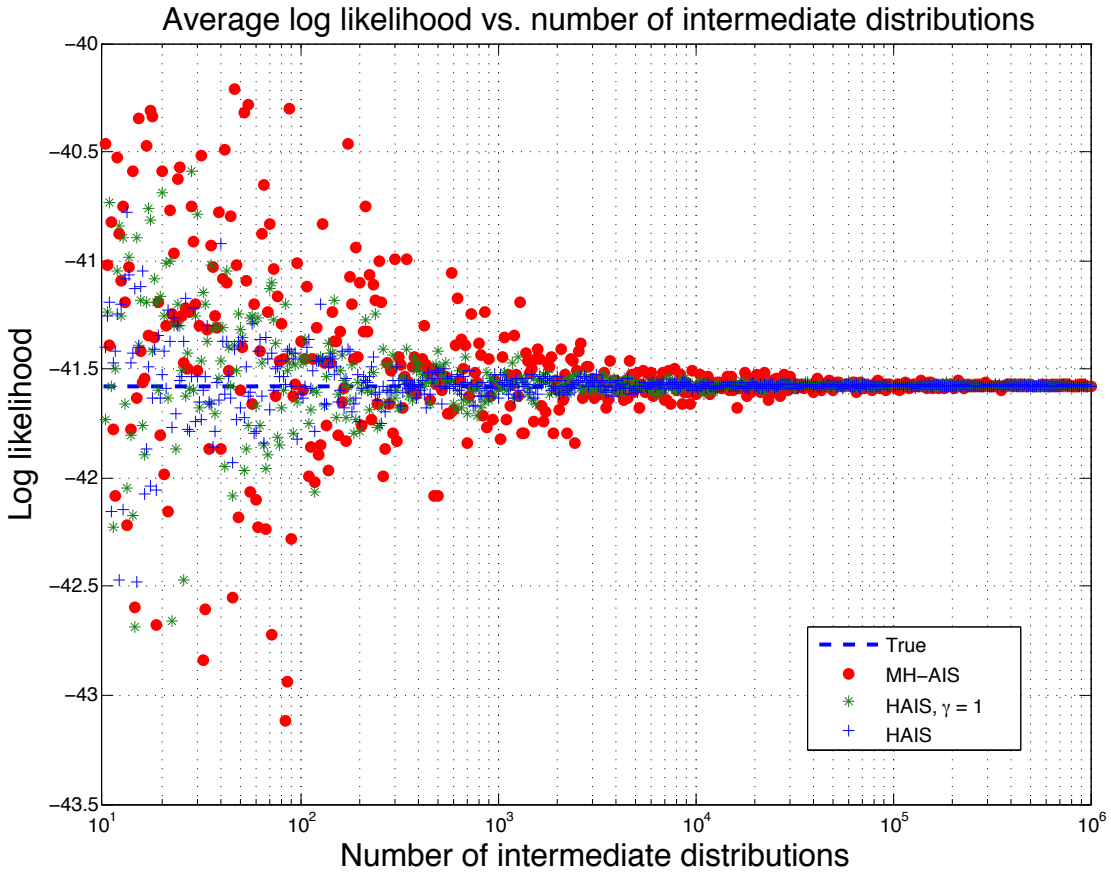


Figure 5.3: Comparison of HAIS with alternate AIS algorithms in a complete ($M = L = 36$) POE Laplace model. Format as in Figure 5.2, but for a Laplace expert.

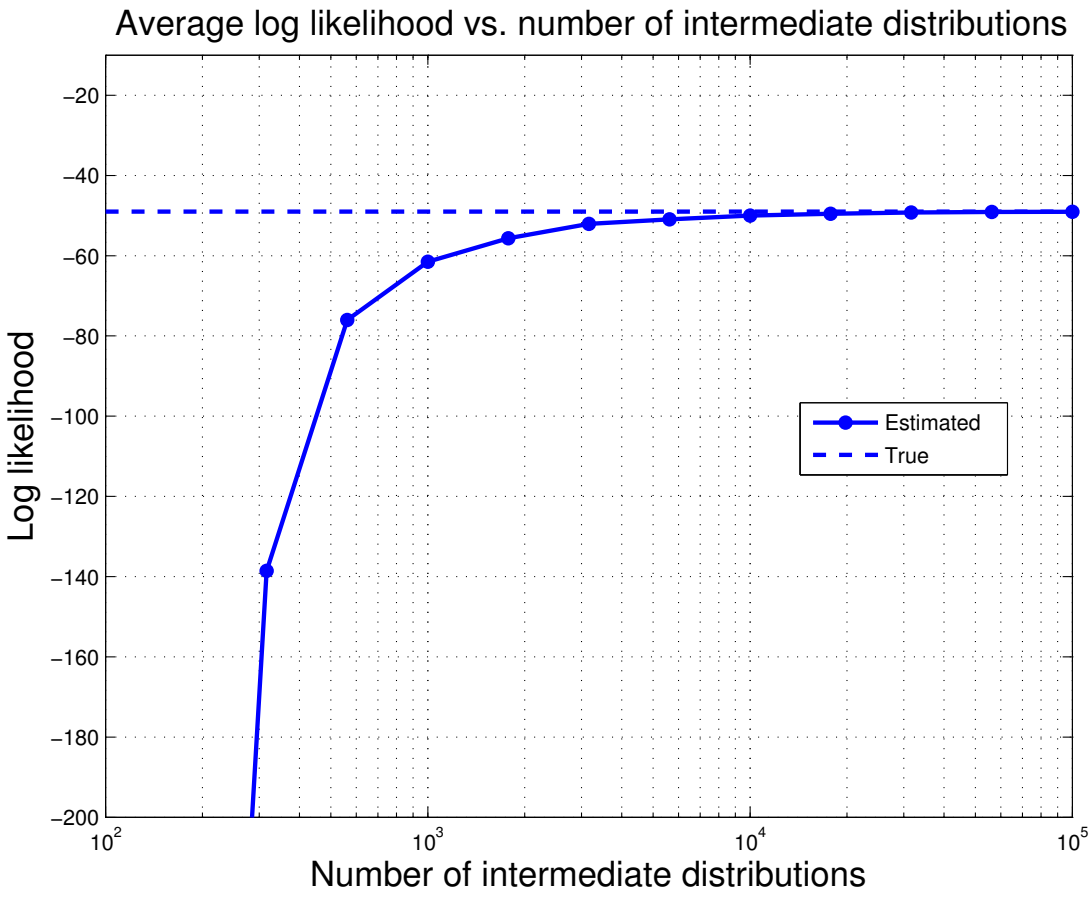


Figure 5.4: Convergence of HAIS for the linear generative model with a Gaussian prior. The dashed blue line indicates the true log likelihood of the test data under the model. The solid blue line indicates the HAIS estimated log likelihood of the test data for different numbers of intermediate distributions N.

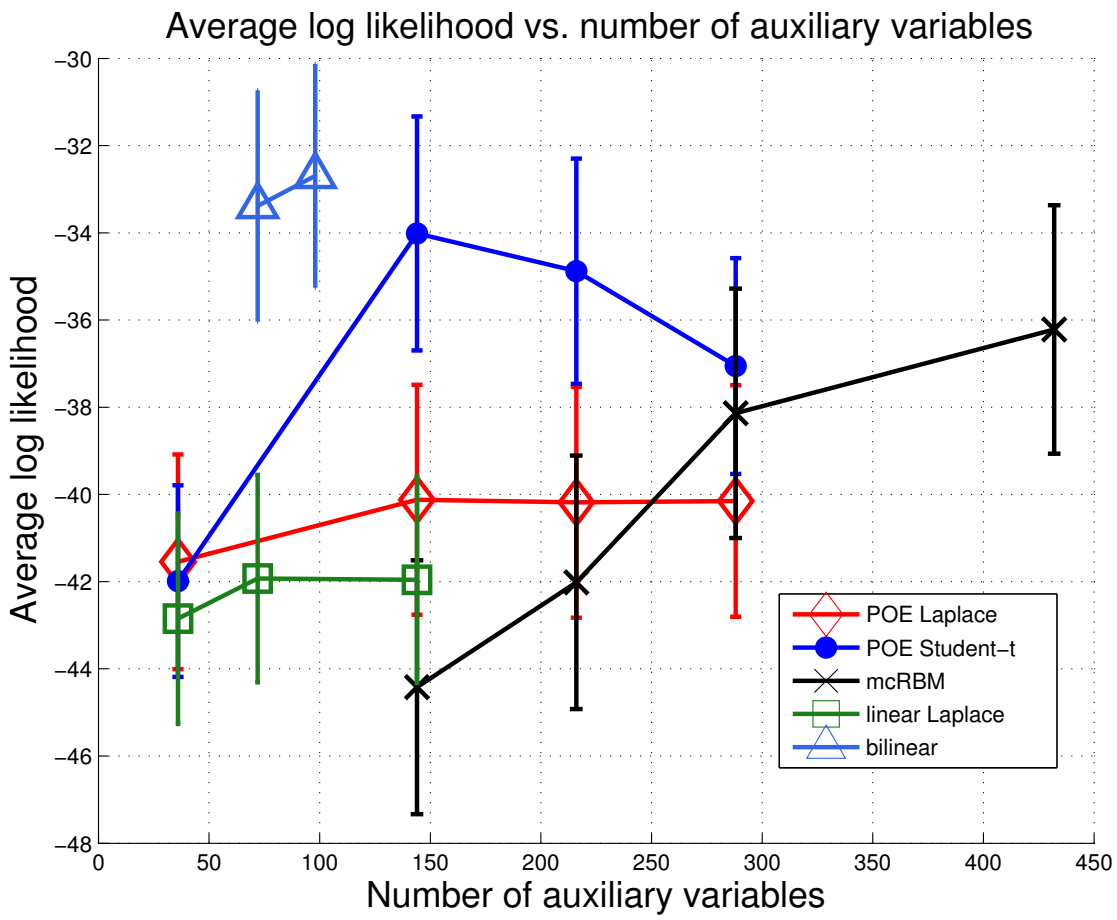


Figure 5.5: Increasing the number of auxiliary variables in a model increases the likelihood it assigns to the test data until it saturates, or overfits.

Chapter 6

Hamiltonian Monte Carlo

Sampling is critical for many tasks involved in learning and working with probabilistic models. As discussed in Section 1.2, Hamiltonian Monte Carlo (HMC) is the current state of the art technique for sampling from high dimensional probabilistic models over continuous state spaces. In this chapter, two extensions to Hamiltonian Monte Carlo which allow more rapid exploration of the state space are presented. Material in this chapter is taken from [Sohl-Dickstein, 2012a].

6.1 Reduced Momentum Flips

Hamiltonian dynamics with partial momentum refreshment, in the style of [Horowitz, 1991], explore the state space more slowly than they otherwise would due to the momentum reversals which occur on proposal rejection. These cause trajectories to double back on themselves, leading to random walk behavior on timescales longer than the typical rejection time, and leading to slower mixing. I present a technique by which the number of momentum reversals can be reduced. This is accomplished by maintaining the net exchange of probability between states with opposite momenta, but reducing the rate of exchange in both directions such that it is 0 in one direction. An experiment illustrates these reduced momentum flips accelerating mixing for a particular distribution.

6.1.1 Formalism

A state $\zeta \in R^{N \times 2}$ consists of a position $\mathbf{x} \in \mathcal{R}^N$ and an auxiliary momentum $\mathbf{v} \in \mathcal{R}^N$, $\zeta = \{\mathbf{x}, \mathbf{v}\}$. The state space has an associated Hamiltonian

$$H(\zeta) = E(\mathbf{x}) + \frac{1}{2}\mathbf{v}^T\mathbf{v}, \tag{6.1}$$

and a joint probability distribution

$$p(\mathbf{x}, \mathbf{v}) = p(\zeta) = \frac{1}{Z} \exp(-H(\zeta)), \tag{6.2}$$

where the normalization constant Z is the partition function.

The momentum flip operator $F : \mathcal{R}^{N\times 2} \to \mathcal{R}^{N\times 2}$ negates the momentum. It has the properties:

- F negates the momentum, $F\zeta = F\{\mathbf{x}, \mathbf{v}\} = \{\mathbf{x}, -\mathbf{v}\}$
- F is its own inverse, $F^{-1} = F$, $FF\zeta = \zeta$.
- F is volume preserving, $\det\left(\frac{\partial(F\zeta)}{\partial\zeta^T}\right) = 1$
- F doesn't change the probability of a state, $p(\zeta) = p(F\zeta)$

The leapfrog integrator $L(n, \epsilon) : \mathcal{R}^{N\times 2} \to \mathcal{R}^{N\times 2}$ integrates Hamiltonian dynamics for the Hamiltonian $H(\zeta)$, using leapfrog integration, for $n \in \mathcal{Z}^+$ integration steps with stepsize $\epsilon \in \mathcal{R}^+$. We assume that n and ϵ are constants, and write this operator simply as L. The leapfrog integrator L has the following relevant properties:

- L is volume preserving, $\det\left(\frac{\partial(L\zeta)}{\partial\zeta^T}\right) = 1$
- L is exactly reversible using momentum flips, $L^{-1} = FLF$, $\zeta = FLFL\zeta$

During sampling, state updates are performed using a transition operator $T(r) : \mathcal{R}^{N\times 2} \to \mathcal{R}^{N\times 2}$, where $r \sim U([0, 1))$ is drawn from the uniform distribution between 0 and 1,

$$T(r)\,\zeta = \begin{cases} L\zeta & r < P_{leap}(\zeta) \\ F\zeta & P_{leap} \le r < P_{leap}(\zeta) + P_{flip}(\zeta) \\ \zeta & P_{leap} + P_{flip}(\zeta) \le r \end{cases}. \tag{6.3}$$

$T(r)$ additionally depends on an acceptance probability for the leapfrog dynamics, $P_{leap}(\zeta) \in [0, 1]$, and a probability of negating the momentum, $P_{flip}(\zeta) \in [0, 1 - P_{leap}(\zeta)]$. These must be chosen to guarantee that $p(\zeta)$ is a fixed point of T.[1]

6.1.2 Making the distribution of interest a fixed point

In order to make $p(\zeta)$ a fixed point, we will choose the Markov dynamics T so that on average as many transitions enter as leave state ζ at equilibrium. This is *not* pairwise

[1] This fixed point requirement can be written as $p(\zeta) = \int d\zeta' p(\zeta') \int_0^1 dr \delta(\zeta - T(r)\,\zeta')$.

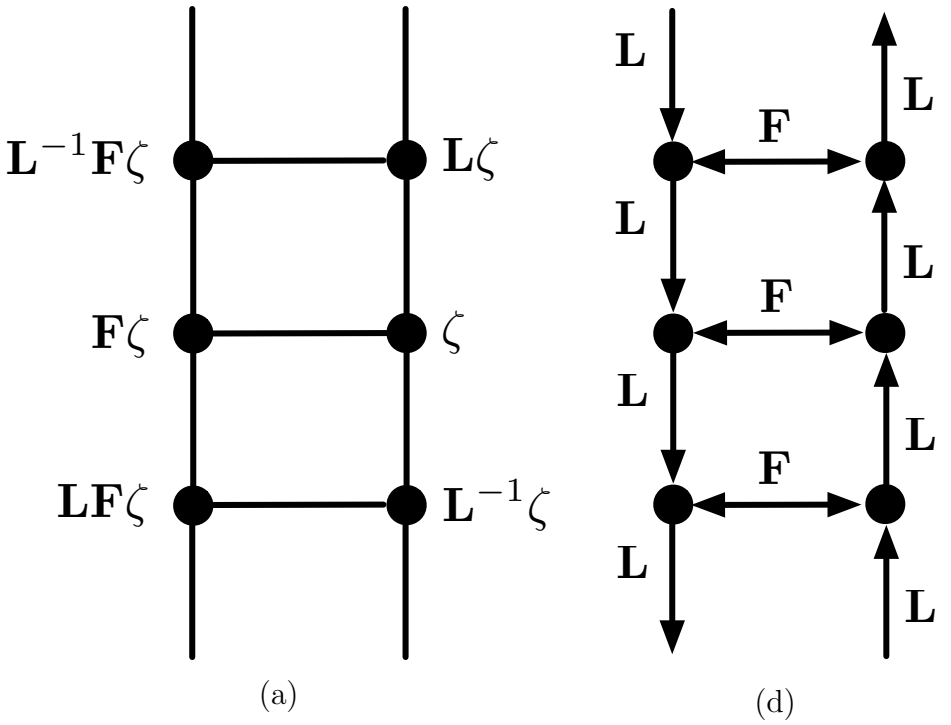


Figure 6.1: This diagram illustrates the possible transitions between states using the Markov transition operator from Equation 6.3. In *(a)* the relevant states, represented by the nodes, are labeled. In *(b)* the possible transitions, represented by the arrows, are labeled. In Section 6.1.2, the net probability flow into and out of the state ζ is set to 0.

detailed balance — instead we are directly enforcing zero net change in the probability of each state by summing over all allowed transitions into or out of the state. This constraint is analogous to Kirchhoff's current law, where the total current entering a node is set to 0. As can be seen from Equation 6.3 and the definitions in Section 6.1.1, and as is illustrated in Figure 6.1, a state ζ can only lose probability to the two states $L\zeta$ and $F\zeta$, and gain probability from the two states $L^{-1}\zeta$ and $F^{-1}\zeta$. Equating the rates of probability inflow and outflow, we find

$$p(\zeta) P_{leap}(\zeta) + p(\zeta) P_{flip}(\zeta) = p\left(L^{-1}\zeta\right) P_{leap}\left(L^{-1}\zeta\right) + p\left(F^{-1}\zeta\right) P_{flip}\left(F^{-1}\zeta\right) \tag{6.4}$$

$$= p\left(L^{-1}\zeta\right) P_{leap}\left(L^{-1}\zeta\right) + p(\zeta) P_{flip}(F\zeta) \tag{6.5}$$

$$P_{flip}(\zeta) - P_{flip}(F\zeta) = \frac{p\left(L^{-1}\zeta\right)}{p(\zeta)} P_{leap}\left(L^{-1}\zeta\right) - P_{leap}(\zeta). \tag{6.6}$$

We choose the standard Metropolis-Hastings acceptance rules for $P_{leap}(\zeta)$,

$$P_{leap}(\zeta) = \min\left(1, \frac{p(L\zeta)}{p(\zeta)}\right). \tag{6.7}$$

Substituting this in to Equation 6.6, we find

$$P_{flip}(\zeta) - P_{flip}(F\zeta) = \frac{p\left(L^{-1}\zeta\right)}{p(\zeta)} \min\left(1, \frac{p\left(LL^{-1}\zeta\right)}{p\left(L^{-1}\zeta\right)}\right) - \min\left(1, \frac{p(L\zeta)}{p(\zeta)}\right) \tag{6.8}$$

$$= \min\left(1, \frac{p\left(L^{-1}\zeta\right)}{p(\zeta)}\right) - \min\left(1, \frac{p(L\zeta)}{p(\zeta)}\right) \tag{6.9}$$

$$= \min\left(1, \frac{p(LF\zeta)}{p(\zeta)}\right) - \min\left(1, \frac{p(L\zeta)}{p(\zeta)}\right). \tag{6.10}$$

Satisfying Equation 6.10 we choose[2] the following form for $P_{flip}(\zeta)$,

$$P_{flip}(\zeta) = \max\left(0, \min\left(1, \frac{p(LF\zeta)}{p(\zeta)}\right) - \min\left(1, \frac{p(L\zeta)}{p(\zeta)}\right)\right). \tag{6.11}$$

Note that $P_{flip}(\zeta) \leq 1 - P_{leap}(\zeta)$, where $1 - P_{leap}(\zeta)$ is the rejection rate, and thus the momentum flip rate, in standard HMC. Using this form for $P_{flip}(\zeta)$ will generally reduce the number of momentum flips required.

[2]To recover standard HMC, instead set $P_{flip}(\zeta) = 1 - P_{leap}(\zeta)$. One can verify by substitution that this satisfies Equation 6.10.

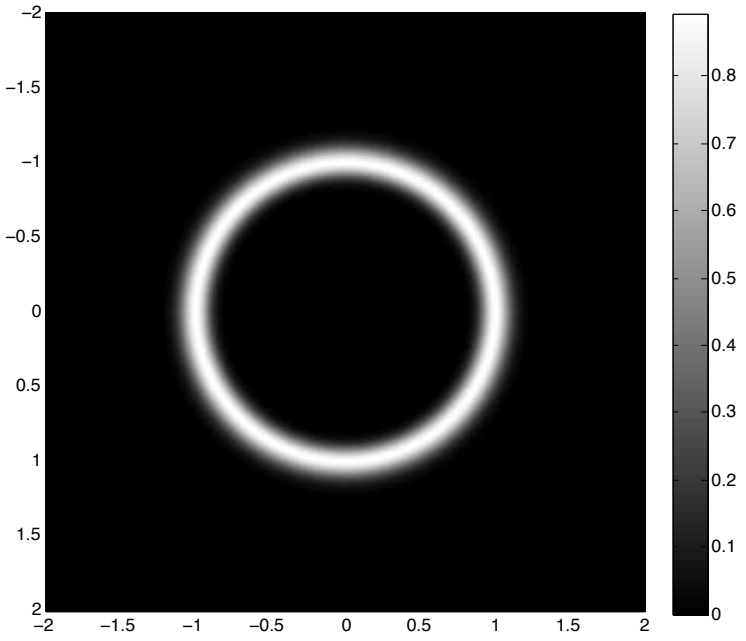


Figure 6.2: A two dimensional image of the distribution used in Section 6.1.3. Pixel intensity corresponds to the probability density function at that location.

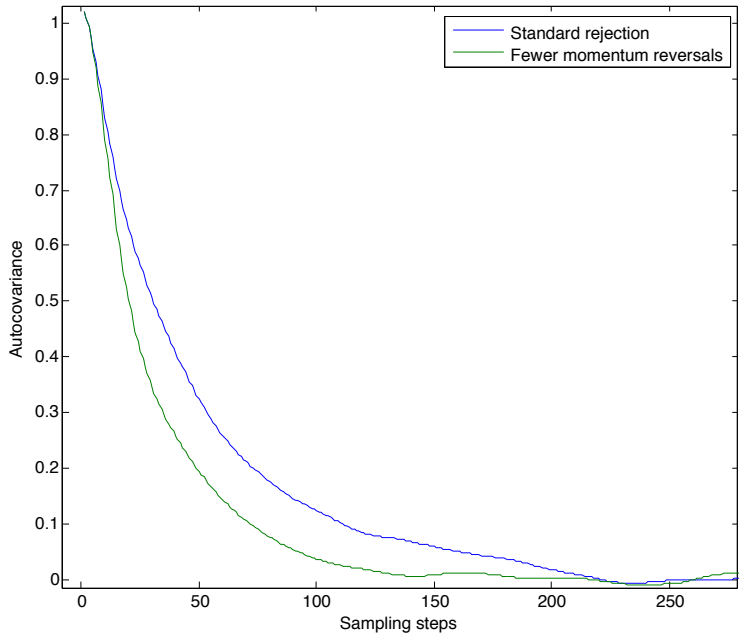


Figure 6.3: The covariance between samples as a function of the number of intervening sampling steps for HMC with standard rejection and rejection with fewer momentum reversals. Reducing the number of momentum reversals causes faster mixing, as evidenced by the faster falloff of the autocovariance.

6.1.3 Example

In order to demonstrate the accelerated mixing provided by this technique, samples were drawn from a simple distribution with standard rejection, and with separate rejection and momentum flipping rates as described above. In both cases, the leapfrog step length ϵ was set to 0.1, the number of integration steps n was set to 1, and the momentum corruption rate β was set so as to corrupt half the momentum per unit stimulation time. Both samplers were run for $100,000$ sampling steps. The distribution used was described by the energy function

$$E = 100 \log^2 \left(\sqrt{x_1^2 + x_2^2} \right). \tag{6.12}$$

A 2 dimensional image of this distribution can be seen in Figure 6.2. The autocovariance of the returned samples can be seen, as a function of the number of intervening sampling steps, in Figure 6.3. Sampling using the technique presented here led to more rapid decay of the autocovariance, consistent with faster mixing.

Chapter 7

Conclusion

Scientific progress is driven by our ability to build models of the world. When investigating complex or large systems, the tools available to build probabilistic models are frequently inadequate. In this thesis I have introduced several new tools that address some of the most pressing issues in probabilistic modeling.

Minimum Probability Flow learning (MPF) is a novel, general purpose framework for parameter estimation in probabilistic models that outperforms current techniques in both learning speed and accuracy. MPF works for any parametric model without hidden state variables, including those over both continuous and discrete state space systems. It avoids explicit calculation of the partition function by employing deterministic dynamics in place of the slow sampling required by many existing approaches. Because MPF provides a simple and well-defined objective function, it can be minimized quickly using existing higher order gradient descent techniques. Furthermore, the objective function is convex for models in the exponential family, ensuring that the global minimum can be found with gradient descent. Extensions to MPF allow it to be used in conjunction with sampling algorithms and persistent particles for even faster performance. Minimum velocity learning, score matching, and some forms of contrastive divergence are special cases of MPF for specific choices for its dynamics.

The natural gradient is a powerful concept, but can be difficult to understand in its traditional presentation. I have made a connection between the natural gradient and the common concept of signal whitening, and additionally provided cookbook techniques for the application of the natural gradient to learning problems. This should lower the barrier to understanding and using this technique in learning problems. Both the natural gradient and MPF allow model fitting to be performed more quickly and accurately, and in situations in which it was previously impractical or impossible.

Hamiltonian Annealed Importance Sampling (HAIS) allows the partition function of non-analytically-normalizable probabilistic models to be computed many times faster than with competing techniques. By improving upon the available methods for partition function estimation, it makes it possible to directly compare large probabilistic models in terms of the

likelihoods they assign to data. This is a fundamental measure of the quality of a model, but one which is very frequently neglected in the literature due to practical and computational limitations. It is my hope that HAIS will lead to more meaningful comparisons between models.

Improvements to Hamiltonian Monte Carlo sampling make many tasks, such as averaging over a distribution, more practical for complex and computationally expensive probabilistic models. I have reduced the time required to generate independent samples from a distribution via Hamiltonian Monte Carlo, by reducing the frequency of momentum flips which cause the sampler to retrace its steps. This will improve the practicality of sampling from a distribution, and lead to more frequent use of samples rather than less accurate approximations or maximum a posteriori estimates.

It is my hope that, taken together, these contributions will improve the ability of scientists and engineers to build and manipulate probabilistic models of the world.

Appendices

Appendix A

Derivation of MPF objective by Taylor expanding KL divergence

The minimum probability flow learning objective function $K(\theta)$ is found by taking up to the first order terms in the Taylor expansion of the KL divergence between the data distribution and the distribution resulting from running the dynamics for a time ϵ:

$$\begin{aligned} K(\theta) &\approx D_{KL}\left(\mathbf{p^{(0)}}||\mathbf{p^{(t)}}(\theta)\right)\Big|_{t=0} \\ &\quad + \epsilon \frac{\partial D_{KL}\left(\mathbf{p^{(0)}}||\mathbf{p^{(t)}}(\theta)\right)}{\partial t}\Big|_{t=0} && \text{(A.1)} \\ &= 0 + \epsilon \frac{\partial D_{KL}\left(\mathbf{p^{(0)}}||\mathbf{p^{(t)}}(\theta)\right)}{\partial t}\Big|_{t=0} && \text{(A.2)} \\ &= \epsilon \frac{\partial}{\partial t}\left(\sum_{i\in\mathcal{D}} p_i^{(0)} \log \frac{p_i^{(0)}}{p_i^{(t)}}\right)\Bigg|_0 && \text{(A.3)} \\ &= -\epsilon \sum_{i\in\mathcal{D}} \frac{p_i^{(0)}}{p_i^{(0)}} \frac{\partial p_i^{(t)}}{\partial t}\Bigg|_0 && \text{(A.4)} \\ &= -\epsilon \sum_{i\in\mathcal{D}} \frac{\partial p_i^{(t)}}{\partial t}\Bigg|_0 && \text{(A.5)} \\ &= -\epsilon \left(\frac{\partial}{\partial t}\sum_{i\in\mathcal{D}} p_i^{(t)}\right)\Bigg|_0 && \text{(A.6)} \\ &= -\epsilon \frac{\partial}{\partial t}\left(1 - \sum_{i\notin\mathcal{D}} p_i^{(t)}\right)\Bigg|_0 && \text{(A.7)} \end{aligned}$$

$$= \epsilon \sum_{i \notin \mathcal{D}} \left. \frac{\partial p_i^{(t)}}{\partial t} \right|_0 \tag{A.8}$$

$$= \epsilon \sum_{i \notin \mathcal{D}} \sum_{j \in \mathcal{D}} \Gamma_{ij} p_j^{(0)} \tag{A.9}$$

$$= \frac{\epsilon}{|\mathcal{D}|} \sum_{i \notin \mathcal{D}} \sum_{j \in \mathcal{D}} \Gamma_{ij}, \tag{A.10}$$

where we used the fact that $\sum_{i \in \mathcal{D}} p_i^{(t)} + \sum_{i \notin \mathcal{D}} p_i^{(t)} = 1$. This implies that the rate of growth of the KL divergence at time $t = 0$ equals the total initial flow of probability from states with data into states without.

Appendix B

Convexity of MPF objective function

As observed by Macke and Gerwinn [Macke and Gerwinn, 2009], the MPF objective function is convex for models in the exponential family.

We wish to minimize

$$K = \sum_{i \in D} \sum_{j \in D^C} \Gamma_{ji} p_i^{(0)}. \tag{B.1}$$

K has derivative

$$\frac{\partial K}{\partial \theta_m} = \sum_{i \in D} \sum_{j \in D^c} \left(\frac{\partial \Gamma_{ij}}{\partial \theta_m} \right) p_i^{(0)} \tag{B.2}$$

$$= \frac{1}{2} \sum_{i \in D} \sum_{j \in D^c} \Gamma_{ij} \left(\frac{\partial E_j}{\partial \theta_m} - \frac{\partial E_i}{\partial \theta_m} \right) p_i^{(0)}, \tag{B.3}$$

and Hessian

$$\begin{aligned} \frac{\partial^2 K}{\partial \theta_m \partial \theta_n} = & \frac{1}{4} \sum_{i \in D} \sum_{j \in D^c} \Gamma_{ij} \left(\frac{\partial E_j}{\partial \theta_m} - \frac{\partial E_i}{\partial \theta_m} \right) \left(\frac{\partial E_j}{\partial \theta_n} - \frac{\partial E_i}{\partial \theta_n} \right) p_i^{(0)} \\ & + \frac{1}{2} \sum_{i \in D} \sum_{j \in D^c} \Gamma_{ij} \left(\frac{\partial^2 E_j}{\partial \theta_m \partial \theta_n} - \frac{\partial^2 E_i}{\partial \theta_m \partial \theta_n} \right) p_i^{(0)}. \end{aligned} \tag{B.4}$$

The first term in the Hessian is a weighted sum of outer products, with non-negative weights $\frac{1}{4} \Gamma_{ij} p_i^{(0)}$, and is thus positive semidefinite. The second term is 0 for models in the exponential family (those with energy functions linear in their parameters).

Parameter estimation for models in the exponential family is therefore convex using minimum probability flow learning.

Appendix C

Lower Bound on Log Likelihood Using MPF

In this appendix, a lower bound on the log probability of data states is derived in terms of the MPF objective function. Although this bound is of theoretical interest, evaluating it requires calculating the first non-zero eigenvalue of the probability flow matrix $\mathbf{\Gamma}$. This is typically an intractable task, and is equivalent to computing the mixing time for a Markov Chain Monte Carlo algorithm.

The probability flow matrix $\mathbf{\Gamma}$ can be written

$$\mathbf{\Gamma} = \mathbf{D} + \mathbf{V}\mathbf{g}\mathbf{V}^{-1}, \tag{C.1}$$

where $\mathbf{V}$ is a diagonal matrix with entries $V_{ii} = \exp\left[-\frac{1}{2}\mathbf{E}_i\right]$, $\mathbf{D}$ is a diagonal matrix with entries $D_{ii} = -\sum_j \left[\mathbf{V}\mathbf{g}\mathbf{V}^{-1}\right]_{ji}$, and $\mathbf{g}$ is the symmetric connectivity matrix.

As observed by Surya Ganguli (personal communication), we can relate $\mathbf{\Gamma}$ to a symmetric matrix $\mathbf{B}$ by an eigenvalue-maintaining transformation,

$$\mathbf{B} = \mathbf{V}^{-1}\mathbf{\Gamma}\mathbf{V} \tag{C.2}$$

$$\mathbf{\Gamma} = \mathbf{V}\mathbf{B}\mathbf{V}^{-1}. \tag{C.3}$$

The eigendecomposition of $\mathbf{B}$ is

$$\mathbf{B} = \mathbf{U}^{\mathbf{B}}\mathbf{\Lambda}\left(\mathbf{U}^{\mathbf{B}}\right)^T, \tag{C.4}$$

where the eigenvalues $\mathbf{\Lambda}$ are identical to the eigenvalues for $\mathbf{\Gamma}$. Single eigenvalues will be written $\lambda_i \equiv \Lambda_{ii}$. The eigenvectors $\mathbf{U}^{\mathbf{B}}$ of $\mathbf{B}$ are orthogonal and taken to be unit length. The

lengths of the eigenvectors for $\mathbf{\Gamma}$ are chosen relative to the eigenvectors for $\mathbf{B}$,

$$\mathbf{U}^{\mathbf{\Gamma}} = \mathbf{V}\mathbf{U}^{\mathbf{B}}. \tag{C.5}$$

Additionally, the eigenvectors and eigenvalues are assumed to be sorted in decreasing order, with $\lambda_1 = 0$, and the corresponding eigenvector of $\mathbf{\Gamma}$, $U^{\Gamma}_{\cdot 1}$, being a scaled version of the model distribution

$$p_i^{(\infty)} = \frac{\exp\left[-E_i\right]}{\sum_j \exp\left[-E_j\right]}, \tag{C.6}$$

with the scaling factor determined by the unit length constraint on $U^{\mathbf{B}}_{\cdot 1}$. The analytic form for the 1st eigenvector of both $\mathbf{\Gamma}$ and $\mathbf{B}$ is,

$$U^{\mathbf{B}}_{i1} = \frac{V_{ii}^{-1} p_i^{(\infty)}}{\left(\sum_j \left(V_{jj}^{-1} p_j^{(\infty)}\right)^2\right)^{\frac{1}{2}}} = \frac{\exp\left[-\frac{1}{2}E_i\right]}{\left(\sum_j \exp\left[-E_j\right]\right)^{\frac{1}{2}}} \tag{C.7}$$

$$U^{\mathbf{\Gamma}}_{i1} = \exp\left[-\frac{1}{2}E_i\right] U^{\mathbf{B}}_{i1} = \frac{\exp\left[-E_i\right]}{\left(\sum_j \exp\left[-E_j\right]\right)^{\frac{1}{2}}}. \tag{C.8}$$

The log probability can then be related to the eigenvector $U^{\mathbf{B}}_{\cdot 1}$,

$$\log p_i^{(\infty)} = \log U^{\mathbf{B}}_{i1} - \frac{1}{2}E_i + \log \frac{\left(\sum_j \exp\left[-E_j\right]\right)^{\frac{1}{2}}}{\sum_j \exp\left[-E_j\right]} \tag{C.9}$$

$$= \log U^{\mathbf{B}}_{i1} + \frac{1}{2}\left(-E_i - \log \sum_j \exp\left[-E_j\right]\right) \tag{C.10}$$

$$= 2 \log U^{\mathbf{B}}_{i1}. \tag{C.11}$$

We now relate the entries in $U^{\mathbf{B}}_{\cdot 1}$ to the initial flow rates $\frac{\partial p_i^{(0)}}{\partial t}$, restricted to data states $i \in \mathcal{D}$. We make the simplifying assumption that the data is sufficiently sparse, and the connectivity matrix $\mathbf{g}$ has been chosen in such a way, that there is no direct flow of probability between data states. Under this assumption, and for data states $i \in \mathcal{D}$,

$$\frac{\partial p_i^{(0)}}{\partial t} = \Gamma_{ii} p_i^{(0)}. \tag{C.12}$$

Noting that Γ can be written $\mathbf{V}\mathbf{U}^{\mathbf{B}}\mathbf{\Lambda}\left(\mathbf{U}^{\mathbf{B}}\right)^{T}\mathbf{V}^{-1}$, we expand Equation C.12,

$$\frac{\partial p_i^{(0)}}{\partial t} = V_{ii}\sum_{j=1}^{N} U_{ij}^{\mathbf{B}}\lambda_j U_{ij}^{\mathbf{B}} V_{ii}^{-1} p_i^{(0)} \tag{C.13}$$

$$= p_i^{(0)}\sum_{j=1}^{N}\left(U_{ij}^{\mathbf{B}}\right)^2\lambda_j. \tag{C.14}$$

Remembering that the eigenvalues are in decreasing order, with $\lambda_1 = 0$ and the remaining eigenvalues negative, and also remembering that $\mathbf{U}^{\mathbf{B}}$ is orthonormal, we write the inequality

$$\frac{\partial p_i^{(0)}}{\partial t} \leq p_i^{(0)}\lambda_2\sum_{j=2}^{N}\left(U_{ij}^{\mathbf{B}}\right)^2 = p_i^{(0)}\lambda_2\left[1-\left(U_{i1}^{\mathbf{B}}\right)^2\right] \tag{C.15}$$

$$U_{i1}^{\mathbf{B}} \geq \left(\max\left[0,\left(1-\frac{\partial p_i^{(0)}}{\partial t}\left(p_i^{(0)}\lambda_2\right)^{-1}\right)\right]\right)^{\frac{1}{2}}. \tag{C.16}$$

Combining Equations C.11 and C.16, for data states $i \in \mathcal{D}$ we find

$$\log p_i^{(\infty)} \geq \log\max\left[0,\left(1-\frac{\partial p_i^{(0)}}{\partial t}\left(p_i^{(0)}\lambda_2\right)^{-1}\right)\right]. \tag{C.17}$$

For sufficiently small magnitude values of $\frac{\partial p_i^{(0)}}{\partial t}\left(p_i^{(0)}\lambda_2\right)^{-1}$ (for instance, because of very small probability flow $\frac{\partial p_i^{(0)}}{\partial t}$, or very negative first non-zero eigenvalue λ_2), we can make the approximation

$$\log\max\left[0,\left(1-\frac{\partial p_i^{(0)}}{\partial t}\left(p_i^{(0)}\lambda_2\right)^{-1}\right)\right] \approx -\frac{\partial p_i^{(0)}}{\partial t}\left(p_i^{(0)}\lambda_2\right)^{-1}. \tag{C.18}$$

The mixing time of a Monte Carlo algorithm with a transition matrix corresponding to $\mathbf{\Gamma}$ can be upper and lower bounded using λ_2. However, λ_2 is generally difficult to find, and frequently little can be said about Monte Carlo mixing times. Equation C.17 can be

rewritten as a bound on λ_2 as follows,

$$p_i^{(\infty)} \geq 1 - \frac{\partial p_i^{(0)}}{\partial t}\left(p_i^{(0)}\lambda_2\right)^{-1} \tag{C.19}$$

$$\lambda_2\left(p_i^{(\infty)} - 1\right) \leq -\frac{\partial p_i^{(0)}}{\partial t}\frac{1}{p_i^{(0)}} \tag{C.20}$$

$$\lambda_2 \geq \frac{\partial p_i^{(0)}}{\partial t}\frac{1}{p_i^{(0)}}\frac{1}{\left(1 - p_i^{(\infty)}\right)} \tag{C.21}$$

$$\frac{\partial p_i^{(0)}}{\partial t} = -\sum_{j\neq i} g_{ji}\exp\left(\frac{1}{2}\left[E_i - E_j\right]\right)p_i^{(0)} \tag{C.22}$$

$$\lambda_2 \geq \frac{-\sum_{j\neq i} g_{ji}\exp\left(\frac{1}{2}\left[E_i - E_j\right]\right)}{\left(1 - p_i^{(\infty)}\right)}. \tag{C.23}$$

$p_i^{(\infty)}$ will typically be much smaller than one, and replacing it with an upper bound will thus have only a small effect on the tightness of the bound in Equation C.23,

$$\lambda_2 \geq \frac{-\sum_{j\neq i} g_{ji}\exp\left(\frac{1}{2}\left[E_i - E_j\right]\right)}{\left(1 - p_i^{(\infty)}\right)}. \geq \frac{-\sum_{j\neq i} g_{ji}\exp\left(\frac{1}{2}\left[E_i - E_j\right]\right)}{\left(1 - p_{i\text{ upper bound}}^{(\infty)}\right)}. \tag{C.24}$$

Equation C.24 holds for any system state i, and the bound on λ_2 can be written in terms of the tightest bound for any system state,

$$\lambda_2 \geq \max_i\left[\frac{-\sum_{j\neq i} g_{ji}\exp\left(\frac{1}{2}\left[E_i - E_j\right]\right)}{\left(1 - p_{i\text{ upper bound}}^{(\infty)}\right)}\right]. \tag{C.25}$$

Appendix D

Score Matching (SM) is a special case of MPF

Score matching, developed by Aapo Hyvärinen [Hyvärinen, 2005], is a method that learns parameters in a probabilistic model using only derivatives of the energy function evaluated over the data distribution (see Equation (D.5)). This sidesteps the need to explicitly sample or integrate over the model distribution. In score matching one minimizes the expected square distance of the score function with respect to spatial coordinates given by the data distribution from the similar score function given by the model distribution. A number of connections have been made between score matching and other learning techniques [Hyvärinen, 2007a; Sohl-Dickstein and Olshausen, 2009; Movellan, 2008a; Lyu, 2009]. Here we show that in the correct limit, MPF also reduces to score matching.

For a d-dimensional, continuous state space, we can write the MPF objective function as

$$\begin{aligned} K_{\mathrm{MPF}} &= \frac{1}{N} \sum_{x \in \mathcal{D}} \int \mathrm{d}^d y \; \Gamma(y, x) \\ &= \frac{1}{N} \sum_{x \in \mathcal{D}} \int \mathrm{d}^d y \; g(y, x) e^{(E(y|\theta) - E(x|\theta))}, \end{aligned} \tag{D.1}$$

where the sum $\sum_{x \in \mathcal{D}}$ is over all data samples, and N is the number of samples in the data set $\mathcal{D}$. Now we assume that transitions are only allowed from states x to states y that are within a hypercube of side length ϵ centered around x in state space. (The master equation will reduce to Gaussian diffusion as $\epsilon \to 0$.) Thus, the function $g(y, x)$ will equal 1 when y is within the x-centered cube (or x within the y-centered cube) and 0 otherwise. Calling this

cube C_ϵ, and writing $y = x + \alpha$ with $\alpha \in C_\epsilon$, we have

$$K_{\mathrm{MPF}} = \frac{1}{N} \sum_{x \in \mathcal{D}} \int_{C_\epsilon} \mathrm{d}^d \alpha \; e^{(E(x+\alpha|\theta) - E(x|\theta))}. \tag{D.2}$$

If we Taylor expand in α to second order and ignore cubic and higher terms, we get

$$\begin{aligned} K_{\mathrm{MPF}} \approx & \frac{1}{N} \sum_{x \in \mathcal{D}} \int_{C_\epsilon} \mathrm{d}^d \alpha \; (1) \\ & - \frac{1}{N} \sum_{x \in \mathcal{D}} \int_{C_\epsilon} \mathrm{d}^d \alpha \; \frac{1}{2} \sum_{i=1}^{d} \alpha_i \nabla_{x_i} E(x|\theta) \\ & + \frac{1}{N} \sum_{x \in \mathcal{D}} \int_{C_\epsilon} \mathrm{d}^d \alpha \; \frac{1}{4} \left(\frac{1}{2} \left[\sum_{i=1}^{d} \alpha_i \nabla_{x_i} E(x|\theta) \right]^2 \right. \\ & \left. - \sum_{i,j=1}^{d} \alpha_i \alpha_j \nabla_{x_i} \nabla_{x_j} E(x|\theta) \right). \end{aligned} \tag{D.3}$$

This reduces to

$$\begin{aligned} K_{\mathrm{MPF}} \approx & \frac{1}{N} \sum_{x \in \mathcal{D}} \left[\epsilon^d + \frac{1}{4} \left(\frac{1}{2} \frac{2}{3} \epsilon^{d+2} \left[\sum_{i=1}^{d} \nabla_{x_i} E(x|\theta) \right]^2 \right. \right. \\ & \left. \left. - \frac{2}{3} \epsilon^{d+2} \nabla_{x_i}^2 E(x|\theta) \right) \right], \end{aligned} \tag{D.4}$$

which, removing a constant offset and scaling factor, is exactly equal to the score matching objective function,

$$K_{\mathrm{MPF}} \sim \frac{1}{N} \sum_{x \in \mathcal{D}} \left[\frac{1}{2} \nabla E(x|\theta) \cdot \nabla E(x|\theta) - \nabla^2 E(x|\theta) \right] \tag{D.5}$$

$$= K_{\mathrm{SM}}. \tag{D.6}$$

Score matching is thus equivalent to MPF when the connectivity function $g(y, x)$ is non-zero only for states infinitesimally close to each other. It should be noted that the score matching estimator has a closed-form solution when the model distribution belongs to the exponential family [Hyvärinen, 2007b], so the same can be said for MPF in this limit.

Appendix E

MPF objective function for an Ising model

This appendix derives the MPF objective function for the case of an Ising model. In Section E.1, connectivity is set between all states which differ by a single bit flip. In Section E.2, an additional connection is included between states which differ in all bits. This additional connection is particularly beneficial in cases (such as spike train data) where unit activity is extremely sparse. Code implementing MPF for the Ising model is available at [Sohl-Dickstein, 2010].

The MPF objective function is

$$K(\mathbf{J}) = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{\mathbf{x}'\notin\mathcal{D}} g(\mathbf{x},\mathbf{x}') \exp\left(\frac{1}{2}\left[E(\mathbf{x};\mathbf{J}) - E(\mathbf{x}';\mathbf{J})\right]\right), \tag{E.1}$$

where $g(\mathbf{x},\mathbf{x}') = g(\mathbf{x}',\mathbf{x}) \in \{0,1\}$ is the connectivity function, $E(\mathbf{x};\mathbf{J})$ is an energy function parameterized by $\mathbf{J}$, and $\mathcal{D}$ is the list of data states. For the Ising model, the energy function is

$$E(\mathbf{x};\mathbf{J}) = \mathbf{x}^T\mathbf{J}\mathbf{x} \tag{E.2}$$

where $\mathbf{x} \in \{0,1\}^N$, $\mathbf{J} \in \mathcal{R}^{N\times N}$, and $\mathbf{J}$ is symmetric ($\mathbf{J} = \mathbf{J}^T$).

E.1 Single Bit Flips

We consider the case where the connectivity function $g(\mathbf{x}, \mathbf{x}')$ is set to connect all states which differ by a single bit flip,

$$g(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \mathbf{x} \text{ and } \mathbf{x}' \text{ differ by a single bit flip,} \quad \sum_n |x_n - x'_n| = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{E.3}$$

The MPF objective function in this case is

$$K(\mathbf{J}) = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n=1}^{N} \exp\left(\frac{1}{2}\left[E(\mathbf{x};\mathbf{J}) - E(\mathbf{x}+\mathbf{d}(\mathbf{x},n);\mathbf{J})\right]\right) \tag{E.4}$$

where the sum over n is a sum over all data dimensions, and the bit flipping function $\mathbf{d}(\mathbf{x}, n) \in \{-1, 0, 1\}^N$ is

$$\mathbf{d}(\mathbf{x}, n)_i = \begin{cases} 0 & i \neq n \\ -(2x_i - 1) & i = n \end{cases} \tag{E.5}$$

For the Ising model, this MPF objective function becomes (using the fact that $\mathbf{J} = \mathbf{J}^T$)

$$K(\mathbf{J}) = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n} \exp\left(\frac{1}{2}\left[\mathbf{x}^T\mathbf{J}\mathbf{x} - (\mathbf{x}+\mathbf{d}(\mathbf{x},n))^T\mathbf{J}(\mathbf{x}+\mathbf{d}(\mathbf{x},n))\right]\right) \tag{E.6}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n} \exp\left(\frac{1}{2}\left[\mathbf{x}^T\mathbf{J}\mathbf{x} - \left(\mathbf{x}^T\mathbf{J}\mathbf{x} + 2\mathbf{x}^T\mathbf{J}\mathbf{d}(\mathbf{x},n) + \mathbf{d}(\mathbf{x},n)^T\mathbf{J}\mathbf{d}(\mathbf{x},n)\right)\right]\right) \tag{E.7}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n} \exp\left(-\frac{1}{2}\left[2\mathbf{x}^T\mathbf{J}\mathbf{d}(\mathbf{x},n) + \mathbf{d}(\mathbf{x},n)^T\mathbf{J}\mathbf{d}(\mathbf{x},n)\right]\right) \tag{E.8}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n} \exp\left(-\frac{1}{2}\left[2\sum_i x_i J_{in}(1-2x_n) + J_{nn}\right]\right) \tag{E.9}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n} \exp\left(\left[(2x_n - 1)\sum_i x_i J_{in} - \frac{1}{2}J_{nn}\right]\right). \tag{E.10}$$

Assume the symmetry constraint on $\mathbf{J}$ is enforced by writing it in terms of another possibly asymmetric matrix $\mathbf{J}' \in \mathcal{R}^{N\times N}$,

$$\mathbf{J} = \frac{1}{2}\mathbf{J}' + \frac{1}{2}\mathbf{J}'^T. \tag{E.11}$$

The derivative of the MPF objective function with respect to $\mathbf{J}'$ is

$$\frac{\partial K\left(\mathbf{J}'\right)}{\partial {J'}_{lm}} = \frac{1}{2}\sum_{\mathbf{x}\in\mathcal{D}} \exp\left(\left[\left(2x_m - 1\right)\sum_i x_i J_{im} - \frac{1}{2}J_{mm}\right]\right)\left[\left(2x_m - 1\right)x_l - \delta_{lm}\frac{1}{2}\right] + \frac{1}{2}\sum_{\mathbf{x}\in\mathcal{D}} \exp\left(\left[\left(2x_l - 1\right)\sum_i x_i J_{il} - \frac{1}{2}J_{ll}\right]\right)\left[\left(2x_l - 1\right)x_m - \delta_{ml}\frac{1}{2}\right], \quad \text{(E.12)}$$

where the second term is simply the first term with indices l and m reversed.

Note that both the objective function and gradient can be calculated using matrix operations (no for loops). See the released Matlab code.

E.2 All Bits Flipped

We consider the case where the connectivity function $g\left(\mathbf{x}, \mathbf{x}'\right)$ is set to connect all states which differ by a single bit flip, and all states which differ in all bits,

$$g\left(\mathbf{x}, \mathbf{x}'\right) = \begin{cases} 1 & \mathbf{x} \text{ and } \mathbf{x}' \text{ differ by a single bit flip,} \quad \sum_n \left|x_n - x'_n\right| = 1 \\ 1 & \mathbf{x} \text{ and } \mathbf{x}' \text{ differ in all bits,} \quad \sum_n \left|x_n - x'_n\right| = N \\ 0 & \text{otherwise} \end{cases}. \quad \text{(E.13)}$$

This extension to the connectivity function aids MPF in assigning the correct relative probabilities between data states and states on the opposite side of the state space from the data, even in cases (such as sparsely active units) where the data lies only in a very small region of the state space.

MPF functions by comparing the relative probabilities of the data states and the states which are connected to the data states. If there is a region of state space in which no data lives, and to which no data states are connected, then MPF is blind to that region of state space, and may assign an incorrect probability to it. This problem has been observed fitting an Ising model to sparsely active neural data. In this case, MPF assigns too much probability to states with many units on simultaneously. However, if an additional connection is added between each state and the state with all the bits flipped, then there are comparison states available which have many units on simultaneously. With this extra connection, MPF better penalizes non-sparse states, and the fit gets much better.

The modified objective function has the form,

$$K\left(\mathbf{J}\right) = K_{single}\left(\mathbf{J}\right) + K_{all}\left(\mathbf{J}\right). \quad \text{(E.14)}$$

We can take the first term, which deals only with single bit flips, from Equation E.10,

$$K_{single}\left(\mathbf{J}\right) = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{n} \exp\left(\left[\left(2x_n - 1\right) \sum_{i} x_i J_{in} - \frac{1}{2} J_{nn}\right]\right). \tag{E.15}$$

The second term is

$$K_{all} = \sum_{\mathbf{x}\in\mathcal{D}} \exp\left(\frac{1}{2}\left[E(\mathbf{x};\mathbf{J}) - E(\mathbf{1}-\mathbf{x};\mathbf{J})\right]\right) \tag{E.16}$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \exp\left(\frac{1}{2}\left[\mathbf{x}^T\mathbf{J}\mathbf{x} - \left(\mathbf{1}-\mathbf{x}\right)^T \mathbf{J}\left(\mathbf{1}-\mathbf{x}\right)\right]\right), \tag{E.17}$$

where $\mathbf{1}$ is the vector of all ones.

The contribution to the derivative from the second term is

$$\frac{\partial K_{all}}{\partial J_{lm}} = \frac{1}{2} \sum_{\mathbf{x}\in\mathcal{D}} \exp\left(\frac{1}{2}\left[\mathbf{x}^T\mathbf{J}\mathbf{x} - \left(\mathbf{1}-\mathbf{x}\right)^T \mathbf{J}\left(\mathbf{1}-\mathbf{x}\right)\right]\right)\left[x_l x_m - \left(1-x_l\right)\left(1-x_m\right)\right]. \tag{E.18}$$

Appendix F

MPF objective function for a Restricted Boltzmann Machine (RBM)

This appendix derives the MPF objective function for the case of a Restricted Boltzmann Machine (RBM), with the connectivity function g_{ij} chosen to connect states which differ by a single bit flip.

The energy function over the visible units for an RBM is found by marginalizing out the hidden units. This gives an energy function of:

$$E(\mathbf{x}) = -\sum_i \log(1 + \exp(-W_i \mathbf{x})) \tag{F.1}$$

where W_i is a vector of coupling parameters and $\mathbf{x}$ is the binary input vector. The MPF objective function for this is

$$K = \sum_{\mathbf{x}\in\mathcal{D}} \sum_n \exp\left(\frac{1}{2}\left[E(\mathbf{x}) - E(\mathbf{x} + \mathbf{d}(\mathbf{x}, n))\right]\right) \tag{F.2}$$

where the sum over n indicates a sum over all data dimensions, and the function $\mathbf{d}(\mathbf{x}, n)$ is

$$\mathbf{d}(\mathbf{x}, n)_i = \begin{cases} 0 & i \neq n \\ -(2x_i - 1) & i = n \end{cases} \tag{F.3}$$

Substituting into the objective function

$$K = \sum_{\mathbf{x}\in\mathcal{D}} \sum_n \exp\left(\frac{1}{2}\left[-\sum_i \log(1 + \exp(-W_i \mathbf{x})) + \sum_i \log(1 + \exp(-W_i \mathbf{x} + W_i \mathbf{d}(\mathbf{x}, n)))\right]\right) \tag{F.4}$$

Matlab code is available at [Sohl-Dickstein, 2010]. It implements the sum over n in a for loop, and calculates the change in $W_i\mathbf{x}$ caused by $W_i\mathbf{d}(\mathbf{x}, n)$ for all samples simultaneously. Note that the for loop could also be performed over samples, with the change induced by each bit flip being calculated by matrix operations. If the code is run with a small batch size, this implementation would be faster. A clever programmer might find a way to replace both for loops with matrix operations.

Bibliography

[Abbey *et al.*, 2009] Craig K. Abbey, Jascha N. Sohl-Dickstein, Bruno A. Olshausen, Miguel P. Eckstein, and John M. Boone. Higher-order scene statistics of breast images. In *Proceedings of SPIE*, volume 7263, pages 726317–726317–10. SPIE, February 2009.

[Ackley *et al.*, 1985] D H Ackley, G E Hinton, and T J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(2):147–169, January 1985.

[Amari and Nagaoka, 2000] SI Amari and H Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. American Mathematical Society, 2000.

[Amari, 1987] Shun-Ichi Amari. *Differential Geometry in Statistical Inference*, volume 10 of *IMS Lecture Notes - Monograph Series*. Inst of Mathematical Statistic, 1987.

[Amari, 1998] Shun-Ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2):251–276, 1998.

[Amari, 2010] Shun-ichi Amari. Information geometry in optimization, machine learning and statistical inference. *Frontiers of Electrical and Electronic Engineering in China*, 5(3):241–260, July 2010.

[Amit *et al.*, 1987] D J Amit, H Gutfreund, and H Sompolinsky. Statistical mechanics of neural networks near saturation. *Annals of Physics*, 173(1):30–67, 1987.

[Aster *et al.*, 2005] R C Aster, B Borchers, and C H Thurber. *Parameter estimation and inverse problems*. Elsevier Academic Press, 2005.

[Bell AJ, 1995] Sejnowski T J Bell AJ. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation 1995; vol. 7:1129-1159*, 1995.

[Bell *et al.*, 2004a] J F Bell, S W Squyres, R E Arvidson, H M Arneson, D Bass, D Blaney, N Cabrol, W Calvin, J Farmer, W H Farrand, W Goetz, M Golombek, J A Grant, R Greeley, E Guinness, A G Hayes, M Y H Hubbard, K E Herkenhoff, M J Johnson, J R

Johnson, J Joseph, K M Kinch, M T Lemmon, R Li, M B Madsen, J N Maki, M Malin, E McCartney, S McLennan, H Y McSween, D W Ming, J E Moersch, R V Morris, E Z Noe Dobrea, T J Parker, J Proton, J W Rice, F Seelos, J Soderblom, L A Soderblom, J N Sohl-Dickstein, R J Sullivan, M J Wolff, and A Wang. Pancam multispectral imaging results from the Spirit Rover at Gusev Crater. *Science (New York, N.Y.)*, 305(5685):800–6, August 2004.

[Bell *et al.*, 2004b] J F Bell, S W Squyres, R E Arvidson, H M Arneson, D Bass, W Calvin, W H Farrand, W Goetz, M Golombek, R Greeley, J Grotzinger, E Guinness, A G Hayes, M Y H Hubbard, K E Herkenhoff, M J Johnson, J R Johnson, J Joseph, K M Kinch, M T Lemmon, R Li, M B Madsen, J N Maki, M Malin, E McCartney, S McLennan, H Y McSween, D W Ming, R V Morris, E Z Noe Dobrea, T J Parker, J Proton, J W Rice, F Seelos, J M Soderblom, L A Soderblom, J N Sohl-Dickstein, R J Sullivan, C M Weitz, and M J Wolff. Pancam multispectral imaging results from the Opportunity Rover at Meridiani Planum. *Science (New York, N.Y.)*, 306(5702):1703–9, December 2004.

[Bell *et al.*, 2006] J. F. Bell, J. Joseph, J. N. Sohl-Dickstein, H. M. Arneson, M. J. Johnson, M. T. Lemmon, and D. Savransky. In-flight calibration and performance of the Mars Exploration Rover Panoramic Camera (Pancam) instruments. *Journal of Geophysical Research*, 111(E2):E02S03, January 2006.

[Besag, 1975] Julian Besag. Statistical Analysis of Non-Lattice Data. *The Statistician, 24(3), 179-195*, 1975.

[Bethge, 2006] Matthias Bethge. Factorial coding of natural images: how effective are linear models in removing higher-order dependencies? *JOSA A*, January 2006.

[Boyd and Vandenberghe, 2004] S P Boyd and L Vandenberghe. *Convex optimization*. Cambridge Univ Press, 2004.

[Broderick *et al.*, 2007] T Broderick, M Dudík, G Tkačik, R Schapire, and W Bialek. Faster solutions of the inverse pairwise Ising problem. *E-print arXiv*, January 2007.

[Brush, 1967] Stephen G. Brush. History of the Lenz-Ising model. *Reviews of Modern Physics*, 39(4):883–893, October 1967.

[Carreira-Perpiñán and Hinton, 2004] M A Carreira-Perpiñán and G E Hinton. On contrastive divergence (CD) learning. *Technical report, Dept. of Computer Science, University of Toronto*, 2004.

[Chandler and Field, 2007] Damon M Chandler and David J Field. Estimates of the information content and dimensionality of natural scenes from proximity distributions. *J Opt Soc Am A Opt Image Sci Vis*, 24(4):922–941, April 2007.

[Chou and Voit, 2009] I C Chou and E O Voit. Recent developments in parameter estimation and structure identification of biochemical and genomic systems. *Math Biosci*, 219:57–83, June 2009.

[Cohen and Grossberg, 1983] M A Cohen and S Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, & Cybernetics*, 1983.

[Cover *et al.*, 1991] T M Cover, J A Thomas, and J Wiley. *Elements of information theory*, volume 1. Wiley Online Library, 1991.

[Cover, 1965] T M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *Electronic Computers, IEEE Transactions on*, (3):326–334, 1965.

[Culpepper *et al.*, 2011] Benjamin J Culpepper, Jascha Sohl-Dickstein, and Bruno A Olshausen. Building a better probabilistic model of images by factorization. *International Conference on Computer Vision*, 2011.

[Gardner, 1987] E Gardner. Maximum storage capacity in neural networks. *EPL (Europhysics Letters)*, 481, 1987.

[Grotzinger *et al.*, 2005] J.P. Grotzinger, R.E. Arvidson, J.F. Bell, W. Calvin, B.C. Clark, D.A. Fike, M. Golombek, R. Greeley, A. Haldemann, K.E. Herkenhoff, B.L. Jolliff, A.H. Knoll, M. Malin, S.M. McLennan, T. Parker, L. Soderblom, J.N. Sohl-Dickstein, S.W. Squyres, N.J. Tosca, and W.A. Watters. Stratigraphy and sedimentology of a dry to wet eolian depositional system, Burns formation, Meridiani Planum, Mars. *Earth and Planetary Science Letters*, 240(1):11–72, November 2005.

[Hayes *et al.*, 2011] A. G. Hayes, J. P. Grotzinger, L. A. Edgar, S. W. Squyres, W. A. Watters, and J. Sohl-Dickstein. Reconstruction of eolian bed forms and paleocurrents from cross-bedded strata at Victoria Crater, Meridiani Planum, Mars. *Journal of Geophysical Research*, 116(E7):E00F21, April 2011.

[Haykin, 2008] S Haykin. *Neural networks and learning machines; 3rd edition*. Prentice Hall, 2008.

[Herkenhoff *et al.*, 2003] KE Herkenhoff, SW Squyres, JF Bell III, JN Maki, HM Arneson, P. Bertelsen, DI Brown, SA Collins, A. Dingizian, ST Elliott, W. Goetz, E. C. Hagerott, A. G. Hayes, M. J. Johnson, R. L. Kirk, S. McLennan, R. V. Morris, L. M. Scherr, M. A. Schwochert, L. R. Shiraishi, G. H. Smith, L. A. Soderblom, J. N. Sohl-Dickstein, and M. V. Wadsworth. Athena Microscopic Imager investigation. *Journal of Geophysical Research*, 108(E12):8065, November 2003.

[Hertz *et al.*, 1991] J Hertz, A Krogh, and R G Palmer. *Introduction to the theory of neural computation*, volume 1. Westview press, 1991.

[Hillar *et al.*, 2012a] C Hillar, N Tran, and K Koepsell. Stable Exponential Storage in Hopfield Networks. 2012.

[Hillar *et al.*, 2012b] Christopher Hillar, Jascha Sohl-Dickstein, and Kilian Koepsell. Efficient and optimal binary Hopfield associative memory storage using minimum probability flow. *arXiv*, 1204.2916, April 2012.

[Hinton and Sejnowski, 1986] G E Hinton and T J Sejnowski. Learning and relearning in Boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:282–317, 1986.

[Hinton *et al.*, 2006] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[Hinton, 2002] G E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[Hopfield, 1982] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554, 1982.

[Horowitz, 1991] A Horowitz. A generalized guided Monte Carlo algorithm. *Physics Letters B*, 268(2):247–252, October 1991.

[Hyvärinen, 2005] A Hyvärinen. Estimation of non-normalized statistical models using score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

[Hyvärinen, 2007a] A Hyvärinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *Computational statistics & data analysis*, 51(5):2499–2512, January 2007.

[Hyvärinen, 2007b] A Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

[Ising, 1925] E Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift fur Physik*, 31:253–258, February 1925.

[Jaakkola and Jordan, 1997] T Jaakkola and M Jordan. A variational approach to Bayesian logistic regression models and their extensions. *Proceedings of the sixth international workshop on artificial intelligence and statistics*, January 1997.

[Jarzynski, 1997] C Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Physical Review E*, January 1997.

[Jinwen, 1993] M Jinwen. The asymmetric Hopfield model for associative memory. *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, 3(1):2611–2614, 1993.

[Johnson *et al.*, 2006] Jeffrey R. Johnson, Jascha Sohl-Dickstein, William M. Grundy, Raymond E. Arvidson, James Bell, Phil Christensen, Trevor Graff, Edward A. Guinness, Kjartan Kinch, Richard Morris, and Michael K. Shepard. Radiative transfer modeling of dust-coated Pancam calibration target materials: Laboratory visible/near-infrared spectrogoniometry. *Journal of Geophysical Research*, 111(E12):E12S07, October 2006.

[Kahn and Marshall, 1953] H Kahn and A Marshall. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1:263–278, January 1953.

[Kappen and Rodriguez, 1997] H Kappen and F Rodriguez. Mean field approach to learning in Boltzmann Machines. *Pattern Recognition Letters*, January 1997.

[Karklin, 2007] Y Karklin. Hierarchical statistical models of computation in the visual cortex. *School of Computer Science, Carnegie Melon University*, Thesis, January 2007.

[Kinch *et al.*, 2007] Kjartan M. Kinch, Jascha Sohl-Dickstein, James F. Bell, Jeffrey R. Johnson, Walter Goetz, and Geoffrey A. Landis. Dust deposition on the Mars Exploration Rover Panoramic Camera (Pancam) calibration targets. *Journal of Geophysical Research*, 112(E6):E06S03, April 2007.

[Little, 1974] WA Little. The existence of persistent states in the brain. *Mathematical Biosciences*, 120:101–120, 1974.

[Lyu, 2009] S Lyu. Interpretation and generalization of Score Matching. *The proceedings of the 25th conference on uncerrtainty in artificial intelligence (UAI*90)*, January 2009.

[Lyu, 2011] Siwei Lyu. Unifying Non-Maximum Likelihood Learning Objectives with Minimum KL Contraction. In J Shawe-Taylor, R S Zemel, P Bartlett, F C N Pereira, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 64–72. 2011.

[MacKay, 2001] D MacKay. Failures of the one-step learning algorithm. *Available electronically at http://www.inference.phy.cam.ac.uk/mackay/abstracts/gbm.html*, January 2001.

[MacKay, 2002] D MacKay. *Information Theory, Inference and Learning Algorithms*. 2002.

[Macke and Gerwinn, 2009] J Macke and S Gerwinn. Personal communication. *Personal communication*, 2009.

[Mahalanobis, 1936] P C Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India*, volume 2, pages 49–55. New Delhi, 1936.

[Marre *et al.*, 2009] O Marre, S El Boustani, Y Fregnac, and A Destexhe. Prediction of spatiotemporal patterns of neural activity from pairwise correlations. *Physical Review Letters*, January 2009.

[McEliece *et al.*, 1987] R McEliece, E Posner, E Rodemich, and S Venkatesh. The capacity of the Hopfield associative memory. *Information Theory, IEEE Transactions on*, 33(4):461–482, 1987.

[Minka, 2005] T Minka. Divergence measures and message passing. *Microsoft Research*, TR-2005-17, January 2005.

[Minsky and Papert, 1988] M Minsky and S Papert. *Perceptrons*. MIT press, 1988.

[Moral *et al.*, 2006] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal Of The Royal Statistical Society*, 68(3):1–26, January 2006.

[Movellan and McClelland, 1993] J R Movellan and J L McClelland. Learning continuous probability distributions with symmetric diffusion networks. *Cognitive Science*, 17:463–496, 1993.

[Movellan, 2008a] J R Movellan. A Minimum Velocity Approach to Learning. *unpublished draft*, January 2008.

[Movellan, 2008b] J R Movellan. Contrastive divergence in Gaussian diffusions. *Neural Computation*, 20(9):2238–2252, 2008.

[Murray and Salakhutdinov, 2009] Iain Murray and Ruslan Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. *Advances in Neural Information Processing Systems*, 21, January 2009.

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

[Neal, 2010] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, January 2010.

[Nocedal, 1980] J Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[Olshausen and Field, 1997] BA A Olshausen and D J Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37(23):3311–3325, December 1997.

[Pathria, 1972] R Pathria. *Statistical Mechanics*. Butterworth Heinemann, January 1972.

[Ranzato and Hinton, 2010] Marc'Aurelio Ranzato and Geoffrey E Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. *IEEE Conference on Computer Vision and Pattern Recognition*, January 2010.

[Rosenblatt, 1957] F Rosenblatt. The perceptron: a perceiving and recognizing automation (projet PARA), Cornell Aeronautical Laboratory Report. 1957.

[Roth and Black, 2005] S Roth and M J Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867. IEEE, 2005.

[Salakhutdinov and Murray, 2008] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. *International Conference on Machine Learning*, 25, January 2008.

[Schmidt, 2005] Mark Schmidt. minFunc. Technical report, http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html, 2005.

[Schneidman *et al.*, 2006] E Schneidman, M J Berry 2nd, R Segev, and W Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012, 2006.

[Shlens *et al.*, 2006] J Shlens, G D Field, J L Gauthier, M I Grivich, D Petrusca, A Sher, A M Litke, and E J Chichilnisky. The structure of multi-neuron firing patterns in primate retina. *Journal of Neuroscience*, 26(32):8254–8266, August 2006.

[Sohl-Dickstein and Culpepper, 2012] Jascha Sohl-Dickstein and Benjamin J. Culpepper. Hamiltonian Annealed Importance Sampling for partition function estimation. *arXiv:1205.1925v1*, May 2012.

[Sohl-Dickstein and Olshausen, 2009] J Sohl-Dickstein and B Olshausen. A Spatial Derivation of score Matching. *Redwood Center Technical Report*, 2009.

[Sohl-Dickstein *et al.*, 2009] J Sohl-Dickstein, P Battaglino, and M DeWeese. Minimum Probability Flow Learning. *arXiv:0906.4779v4*, January 2009.

[Sohl-Dickstein *et al.*, 2010] Jascha Sohl-Dickstein, Jimmy C. Wang, and Bruno A. Olshausen. An Unsupervised Algorithm For Learning Lie Group Transformations. *arXiv:1001.1027v3*, January 2010.

[Sohl-Dickstein *et al.*, 2011a] Jascha Sohl-Dickstein, Peter Battaglino, and Michael DeWeese. New Method for Parameter Estimation in Probabilistic Models: Minimum Probability Flow. *Physical Review Letters*, 107(22):11–14, November 2011.

[Sohl-Dickstein *et al.*, 2011b] Jascha Sohl-Dickstein, Peter B. Battaglino, and Michael R. DeWeese. Minimum Probability Flow Learning. *International Conference on Machine Learning*, 107(22):11–14, November 2011.

[Sohl-Dickstein, 2010] Jascha Sohl-Dickstein. http://github.com/Sohl-Dickstein/Minimum-Probability-Flow-Learning, 2010.

[Sohl-Dickstein, 2011] Jascha Sohl-Dickstein. http://github.com/Sohl-Dickstein/Hamiltonian-Annealed-Importance-Sampling, 2011.

[Sohl-Dickstein, 2012a] Jascha Sohl-Dickstein. Hamiltonian Monte Carlo with Reduced Momentum Flips. *arXiv:1205.1939v1*, May 2012.

[Sohl-Dickstein, 2012b] Jascha Sohl-Dickstein. The Natural Gradient by Analogy to Signal Whitening, and Recipes and Tricks for its Use. *arXiv:1205.1828v1*, May 2012.

[Sommer and Dayan, 1998] F T Sommer and P Dayan. Bayesian retrieval in associative memories with storage errors. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 9(4):705–713, January 1998.

[Stephens *et al.*, 2008] Greg J Stephens, Thierry Mora, Gasper Tkacik, and William Bialek. Thermodynamics of natural images. *Arxiv preprint arXiv:0806.2694*, January 2008.

[Swendsen and Wang, 1987] R H Swendsen and J S Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58(2):86–88, 1987.

[Tanaka, 1998] T Tanaka. Mean-field theory of Boltzmann machine learning. *Physical Review Letters E*, January 1998.

[Tang *et al.*, 2008] A Tang, D Jackson, J Hobbs, Wei Chen, Jodi L Smith, Hema Patel, Anita Prieto, Dumitru Petrusca, Matthew I Grivich, A Sher, Pawel Hottowy, Wladyslaw Dabrowski, Alan M Litke, and John M Beggs. A maximum entropy model applied to spatial and temporal correlations from cortical networks in vitro. *Journal of Neuroscience*, January 2008.

[Theis, 2005] FJ Theis. Gradients on matrix manifolds and their chain rule. *Neural Information Processing-Letters and Reviews*, 2005.

[Tieleman, 2008] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. *Proceedings of the 25th international conference on*, pages 1064–1071, 2008.

[van Hateren and van der Schaaf, 1998] J H van Hateren and A van der Schaaf. Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex. *Proceedings: Biological Sciences*, 265(1394):359–366, March 1998.

[Wang *et al.*, 2011] C M Wang, J Sohl-Dickstein, I Tosic, and B A Olshausen. Lie Group Transformation Models for Predictive Video Coding. *Data Compression Conference (DCC), 2011*, pages 83–92, 2011.

[Weisbuch and Fogelman-Soulié, 1985] G Weisbuch and F Fogelman-Soulié. Scaling laws for the attractors of Hopfield networks. *Journal de Physique Lettres*, 46(14):623–630, 1985.

[Welling and Hinton, 2002] M Welling and G Hinton. A new learning algorithm for mean field Boltzmann machines. *Lecture Notes in Computer Science*, January 2002.

[Yu *et al.*, 2008] S Yu, D Huang, W Singer, and D Nikolic. A small world of neuronal synchrony. *Cerebral Cortex*, January 2008.

[Yuille, 2005] A Yuille. The Convergence of Contrastive Divergences. *Department of Statistics, UCLA. Department of Statistics Papers.*, 2005.

[Zoran and Weiss, 2009] Daniel Zoran and Yair Weiss. The "Tree-Dependent Components" of Natural Images are Edge Filters. *Neural and Information Processing Systems*, January 2009.

[Zweig, 1998] G Zweig. Speech recognition with dynamic Bayesian networks. 1998.