Minimum Probability Flow Learning:  
A New Method For Fitting Probabilistic Models

by

Peter Brian Battaglino

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

Doctor of Philosophy

in

Physics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael Deweese, Chair
Professor Joel Fajans
Professor Bruno Olshausen

Fall 2014
Minimum Probability Flow Learning:
A New Method For Fitting Probabilistic Models

Copyright © 2014

by

Peter Brian Battaglino
Abstract

Minimum Probability Flow Learning: A New Method For Fitting Probabilistic Models
by
Peter Brian Battaglino
Doctor of Philosophy in Physics
University of California, Berkeley
Professor Michael Deweese, Chair

Fitting probabilistic models to data is often difficult, due to the general intractability of the partition function and its derivatives. In this dissertation we propose a new parameter estimation technique that does not require computing an intractable normalization factor or sampling from the equilibrium distribution of the model. This is achieved by establishing dynamics that would transform the observed data distribution into the model distribution, and then setting as the objective the minimization of the KL divergence between the data distribution and the distribution produced by running the dynamics for an infinitesimal time. Score matching, minimum velocity learning, and certain forms of contrastive divergence are shown to be special cases of this learning technique. We demonstrate parameter estimation in Ising models, deep belief networks and an independent component analysis model of natural scenes. In the Ising model case, current state of the art techniques are outperformed by at least an order of magnitude in learning time, with lower error in recovered coupling parameters.
Acknowledgements

First and foremost, I would like to thank Jascha Sohl-Dickstein, the other half of this work, without whom it would have wilted. Equally, I would like to thank my advisor Mike Deweese, for being everything an advisor should be, and so much more — a source of encouragement, perspective, inspiration, and a bully that made me turn the crank when I didn’t want to. Thanks Mike! Thanks to Ashley for being patient and positive and providing a continuous supply of beans, and thanks of course to Mom, Dad, and Casey.
Dedicated to Grandma Olga
Contents

1 Introduction

1.1 Learning Probabilistic Models

1.1.1 On the Ever-Growing Complexity of Data Sets

1.1.2 The Problem of Parameter Fitting

1.1.3 Maximum Likelihood Learning

1.1.4 Aside: Maximum Likelihood Estimators and the Cramer-Rao Bound

1.1.5 Example: parameter estimation for visible Boltzmann machines

1.1.6 When Maximum Likelihood Fails

1.1.7 Sampling techniques

1.2 Alternatives to Maximum Likelihood

1.2.1 Contrastive Divergence

1.2.2 Score Matching

1.2.3 Maximum Pseudolikelihood Learning

1.2.4 Minimum Velocity Learning

2 Minimum Probability Flow Learning

2.1 Saying Goodbye to the Partition Function

2.2 Prelude: Lifting Linear Diffusion Processes to a General State Space

2.3 The Minimum Probability Flow Learning Algorithm

2.3.1 Distributions

2.3.2 Dynamics

2.3.3 Detailed Balance

2.3.4 Objective Function

2.3.5 Tractability

2.3.6 Continuous State Spaces

2.3.7 Choosing the Connectivity Function $g$

2.3.8 Sampling the Connectivity Matrix $\Gamma$
Chapter 1

Introduction

This dissertation addresses the problem of fitting probabilistic models to data. In particular, the focus is on probabilistic models for which computing individual model probabilities is not feasible, due to the intractability of the partition function. After providing some background in the basics of parameter fitting, we will approach this problem by first conducting a short survey of existing techniques, along with their strengths and weaknesses, and we will subsequently unify them under a new framework that we call minimum probability flow learning. While this framework is quite general, we will show that it offers a very simple and specific prescription for finding best fit parameters, and that the achieved results are in many cases of higher quality, and converge faster, than those yielded by existing techniques. A large portion of the work described in this dissertation was jointly carried out by myself, Jascha Sohl-Dickstein, and my advisor Michael Deweese, and has been published under [35; 37].

1.1 Learning Probabilistic Models

1.1.1 On the Ever-Growing Complexity of Data Sets

Larger, more complicated datasets are constantly emerging in diverse areas of the natural sciences. In the field of neuroscience, the ability to measure spiking data simultaneously from hundreds of cortical neurons [10] is increasing the need for efficient probabilistic modeling techniques that are tractable over high-dimensional datasets. Fitting well-established probabilistic models from physics to population neural activity recorded in retina [32; 33; 32] or cortex [40; 24; 41] is currently impractical for populations of more than about 100 neurons [5]. Genetic microarray readings [20] are producing the same need among computational biologists. Understanding the statistics of natural scenes [34] is viewed as crucial to understanding the human
visual system, but generative models of natural images still leave much to be desired.

One of the main limiting factors that all of these areas experience is the computational intractability of parameter fitting. As the complexity of data increases, more and more of that complexity is lost on a simple, easy-to-fit model, and it becomes necessary to bring out bigger and more sophisticated analytical tools to deal with that complexity. It’s (relatively) easy to come up with an all-purpose, rich family of models that could, in theory, capture the nuances of a rich, high-dimensional dataset. The problem one encounters is that of exploring the parameter spaces of those models sufficiently to match the specific data in question.

Parameter search spaces are often plagued with local extrema, and even when not, the task of even evaluating goodness of fit using standard methods like maximum likelihood can be giant using today’s available computing power. Even evaluating the probability of a single state under a model can prove practically impossible due to the difficulty of computing a model’s normalization factor given its parameters. Because this computation is exponentially hard in the number of dimensions of the state space, this limitation will always make calculation of probabilities impractical in sufficiently high dimensions.

It is for this reason that a significant fraction of the field of statistical learning theory is devoted to the tasks of evaluating the efficiency of estimation techniques and producing faster, more computationally tractable methods for parameter estimation.

1.1.2 The Problem of Parameter Fitting

Parameter estimation can be viewed as the inverse of the usual problem physicists face: rather than assuming fixed model parameters, such as coupling strengths in an Ising spin glass, and then predicting observable properties of the system, such as spin-spin correlations, our goal is to start with a series of observations and then estimate the underlying model parameters. The observations will in general consist of features \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), where each \( x_i \) can be either continuous or discrete. A dataset consists of a collection of \( N \) such feature vectors, which we will call \( \mathcal{D} \). We wish to fit a particular probabilistic model \( p(\mathbf{x}|\theta) \), chosen from a family of models parametrized by the parameter vector \( \theta \), in such a way that the model best summarizes the statistics of the observed dataset \( \mathcal{D} \). The result is an estimator \( \hat{\theta} \) for the parameters \( \theta \). The quality of such an estimator can be evaluated using its variance, and the deviation of its mean from the values specified by \( \theta \). In addition, it is important to consider the computational resources required to evaluate such an estimator, and whether \( \hat{\theta} \) can be faithfully evaluated at all in a reasonable amount of time using the available dataset.

A common example of a simple parameter estimation problem is that of fitting a Gaussian curve to normally distributed data, as in Figure 1.1. For low dimensional
Chapter 1. Introduction

Figure 1.1: A Gaussian distribution (curve) fit to normally distributed data (histogram)

feature spaces and simple model classes like the Gaussian example, this problem has been thoroughly analyzed, but as the dimensionality \( d \) of a feature space and the complexity of the desired model increases, it becomes increasingly difficult. Many powerful models are regarded more as interesting theoretical structures rather than as versatile analytical tools due to the difficulty they present when trying to fit them to large datasets in finite time, an example being Boltzmann machines \([1, 18, 39]\). This often highlights the inadequacy of the available fitting methods rather than the choice of an overly complex model. For example, natural images form an incredibly nuanced body of data that warrants the use of powerful, complex models in order to capture enough of the higher order statistics present. Boltzmann machines (see Figure 1.2) seem a likely candidate for being able to accurately capture adequate details in the space of natural images, but so far, only restricted classes of Boltzmann machines have been applied to the task with any degree of success \([31]\).

For a probabilistic model \( p(x|\theta) \), we will adopt the notation

\[
p(x|\theta) = \frac{1}{Z(\theta)} \exp[-E(x; \theta)],
\]

where

\[
Z(\theta) = \sum_x \exp[-E(x; \theta)]
\]

is a normalization factor commonly called the partition function. These types of models are often called “energy based models”, and the function \( E(x; \theta) \) is called an
Chapter 1. Introduction

Figure 1.2: A section of a general Boltzmann machine with both observed and latent nodes. A Boltzmann machine is an energy-based model with a distribution of the form $p(x|W) = \exp[-\sum_{ij} W_{ij} x_i x_j]/Z(W)$, where $x = (v, h)^T$.

“energy”. It is perhaps misleading to think of energy based models as a separate, or more specific class of models. Any model that you can write in the form $p(x|\theta)$ can also be written in the form of equation 1.1 by noting that

$$p(x|\theta) = \exp[\log p(x|\theta)]$$

$$= \frac{\exp[\log p(x|\theta) + \log Z(\theta)]}{Z(\theta)},$$

so we can identify

$$E(x; \theta) = -\log p(x|\theta) - \log Z(\theta).$$

Note that there is, however, an extra degree of freedom, because any real constant can be added to the energy function as long as it is accompanied by a corresponding constant multiplicative transformation of $Z(\theta)$.

1.1.3 Maximum Likelihood Learning

The de facto method for fitting a probabilistic model to data is called maximum likelihood learning (ML). This approach involves maximizing the (log) likelihood of the model parameters given the observed data with respect to the model parameters. Given a list of $N$ observed data samples $\mathcal{D}$, defined by

$$\mathcal{D} = \{x_i \mid i = 1 \ldots N\},$$

4
we define the log likelihood as

$$\ell(\theta, D) = \log p(D|\theta).$$

(1.7)

In this report, we will assume unless otherwise stated that the list of observed data $D$ is independent and identically distributed (i.i.d.), that is

$$p(D|\theta) = \prod_{i=1}^{N} p_X(x_i|\theta).$$

(1.8)

This assumption enables us to treat the log likelihood using the simplified form

$$\ell(\theta, D) = \sum_{i=1}^{N} \ell(\theta, x_i),$$

where

$$\ell(\theta, x_i) = \log p_X(x_i|\theta).$$

(1.10)

It is always possible to write the probability mass function $p_X(x|\theta)$ in the form

$$p_X(x|\theta) = \frac{1}{Z(\theta)} e^{-E(x;\theta)},$$

(1.11)

where we refer to the function

$$Z(\theta) = \sum_{\text{all } x} e^{-E(x;\theta)}$$

(1.12)

as the “partition function”, or normalization factor of the model, and the function $E(x;\theta)$ as the “energy”. These terms are borrowed from the statistical physics literature [29]. In this notation, we can write the likelihood function $\ell(\theta, D)$ as

$$\ell(\theta, D) = -\sum_{i=1}^{N} E(x_i;\theta) - N \log Z(\theta).$$

(1.13)

The maximum likelihood estimator is evaluated by finding parameters $\theta$ that maximize the function $\ell(\theta, D)$.

**Example 1.1.1** (Gaussian distribution). A Gaussian distribution for a $d$-dimensional
random variable $X$ with mean $\mu$ and covariance matrix $\Sigma$ has the form

$$\text{Prob}(X = x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \det \Sigma^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}.$$  \hspace{1cm} (1.14)

For this example, the energy function is

$$E(x; \mu, \Sigma) = \frac{1}{2} (x - \mu)^T \Sigma^{-1}(x - \mu),$$ \hspace{1cm} (1.15)

and the partition function is

$$Z(\mu, \Sigma) = [(2\pi)^d \det \Sigma]^{1/2}.$$ \hspace{1cm} (1.16)

Given a dataset $D = \{x_i | i = 1 \ldots N\}$, the log likelihood is

$$\ell = -\frac{N}{2} \left( \log(2\pi)^d + \log \det \Sigma + \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \right).$$ \hspace{1cm} (1.17)

Setting $\partial \ell / \partial \mu = 0$ gives

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i,$$ \hspace{1cm} (1.18)

and setting $\partial \ell / \partial \Sigma = 0$ gives

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T,$$ \hspace{1cm} (1.19)

where the formulas

$$\frac{\partial \det A}{\partial A_{mn}} = (\det A)(A^{-1})_{mn}$$ \hspace{1cm} (1.20)

and

$$\frac{\partial (A^{-1})_{ij}}{\partial A_{mn}} = -(A^{-1})_{im}(A^{-1})_{nj},$$ \hspace{1cm} (1.21)

and the fact that $\Sigma$ is a symmetric matrix were used. In this case, the maximum likelihood solutions for the parameters $\Sigma$ and $\mu$ are just the sample covariance and sample mean, respectively.
In situations where it is impossible or unwieldy to derive an explicit solution for the estimate \( \hat{\theta}_{\text{ML}} \) (called the ML estimator) that maximizes the likelihood function \( \ell \), it is common to perform gradient ascent to find the maximum. As we will soon see, it is more often the case than not that there are problems with gradient ascent approaches as well. To perform gradient ascent to find a maximum of an objective function (in this case \( \ell(\theta, D) \)), we compute the gradient of the objective function and travel a small step in the direction given by that gradient. If we perform this action enough times and with a small enough step size, we will eventually reach a local maximum of the objective function. Note that there is no guarantee that the maximum reached will be global unless the objective function is convex. Consider the gradient of \( \ell \):

\[
\frac{\partial \ell(\theta, D)}{\partial \theta} = -\sum_{i=1}^{N} \frac{\partial E(x_i; \theta)}{\partial \theta} + N \frac{\partial \log Z(\theta)}{\partial \theta}
\]

\[
= -N \left[ \frac{1}{N} \sum_{i=1}^{N} \frac{\partial E(x_i; \theta)}{\partial \theta} - \frac{\partial}{\partial \theta} \sum_{\text{all } x} e^{-E(x; \theta)} \right]
\]

\[
= -N \left[ \frac{1}{N} \sum_{i=1}^{N} \frac{\partial E(x_i; \theta)}{\partial \theta} - \sum_{\text{all } x} \frac{\partial E(x; \theta)}{\partial \theta} e^{-E(x; \theta)} \right]
\]

\[
= -N \left[ \langle \frac{\partial E(x; \theta)}{\partial \theta} \rangle_{\text{data}} - \langle \frac{\partial E(x; \theta)}{\partial \theta} \rangle_{\text{model}} \right].
\]  

Thus \( \ell \) is maximized when the expected \( \theta \) gradient of the energy over the data distribution is equal to the expected \( \theta \) gradient of the energy over the model distribution.

Define the Kullback-Leibler (KL) divergence \[11\] between two probability distributions \( p(x) \) and \( q(x) \) defined over the same state space as

\[
D_{\text{KL}}(p \parallel q) = \sum_{\text{all } x} p(x) \log \frac{p(x)}{q(x)}.
\]  

Thinking of the data distribution of \( D \) as the theoretical distribution underlying \( D \), one sampling of which is \( D \) (and denoting it by \( p_{\text{data}}(x) \)), consider the KL divergence between the data distribution of \( D \) and the model distribution \( p_X(x|\theta) \). Its negative
gradient with respect to $\theta$ is

$$
- \frac{\partial}{\partial \theta} D_{KL} \left( p_{\text{data}}(x) \| p_{X}(x|\theta) \right) = - \frac{\partial}{\partial \theta} \sum_{x} p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{X}(x|\theta)}
$$

$$
= \sum_{x} p_{\text{data}}(x) \frac{\partial}{\partial \theta} \log p_{X}(x|\theta)
$$

$$
= - N \left[ \frac{1}{N} \sum_{i=1}^{N} \frac{\partial E(x_i; \theta)}{\partial \theta} - \frac{\partial}{\partial \theta} \sum_{x} e^{-E(x;\theta)} \right],
$$

(1.25)

which is the same gradient as in equation 1.22. Thus we see that minimizing the KL divergence between the distributions of the data and the model with respect to the parameters $\theta$ is equivalent to maximizing the (log) likelihood function of the model parameters given the dataset $\mathcal{D}$:

$$
\hat{\theta}_{ML} = \arg\min_{\theta} D_{KL} \left( p_{\text{data}}(x) \| p_{X}(x|\theta) \right).
$$

(1.26)

Thinking of ML estimation in this way will be helpful for understanding how Minimum Probability Flow estimation works later.

### 1.1.4 Aside: Maximum Likelihood Estimators and the Cramér-Rao Bound

If we wish to estimate some statistic of our model over a sample, we generally wish our estimator to be unbiased and low-variance. The ML estimator obeys the nice property that it saturates the Cramér-Rao bound [12; 30] asymptotically (in the limit of large sample size). The Fisher information [11] of a probabilistic model is defined as

$$
I_{ij} = \mathbb{E} \left[ \frac{\partial^2 \ell(\theta, \mathcal{D})}{\partial \theta_i \partial \theta_j} \right].
$$

(1.27)

The Cramér-Rao theorem states that the covariance of an unbiased estimator $\hat{\theta}$ obeys the relation

$$
cov(\hat{\theta}) \geq I^{-1},
$$

(1.28)

where we say matrices $A$ and $B$ obey $A \geq B$ if the matrix $A - B$ is positive semi-definite. When evaluating other possible estimators $\hat{\theta}$, it is a good sign if you come close to saturating the Cramér-Rao bound.
1.1.5 Example: parameter estimation for visible Boltzmann machines

A visible Boltzmann machine \[139\] is a type of probabilistic model over a binary state space that has \( v \) visible units \( x = (x_1, x_2, \ldots, x_v) \) encoding a data vector, where each \( x_i \in \{0, 1\} \). Figure 1.3 shows a Boltzmann machine with three visible units as an undirected graph. Nodes in the graph represent binary units, and edges represent interaction energies between nodes. The contribution of an edge \((ij)\) is the energy \(W_{ij}x_i x_j\). Note that an edge only contributes a non-zero energy if both of its corresponding nodes are “on”. Self interactions \((ii)\) contribute an energy \(W_{ii}x_i^2 = W_{ii}x_i\).

Example 1.1.2 (A visible Boltzmann machine with only one unit). The simplest visible Boltzmann machine that can be constructed has only one node, as shown in Figure 1.4. The energy function for this Boltzmann machine has the form

\[
E(x; W) = W x, \tag{1.29}
\]

giving a probability distribution

\[
p(x|W) = \frac{\exp(-W x)}{1 + \exp(-W)}. \tag{1.30}
\]

This is of course just the distribution of a Bernoulli random variable, parameterized
in terms of an energy function, with off and on probabilities given by

\[
p(X = 0|W) = \frac{1}{1 + \exp(-W)} := p_0 \tag{1.31}
\]

\[
p(X = 1|W) = \frac{1}{1 + \exp(+W)} := p_1 = 1 - p_0, \tag{1.32}
\]

respectively.

What does the maximum likelihood solution look like in this case? Suppose our collection of observed data \( \mathcal{D} \) looks like

\[
\mathcal{D} = \{0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, \ldots\}, \tag{1.33}
\]

where \( X \) takes on the value 1 a fraction \( f \) of the time, and it takes on the value 0 a fraction \( 1 - f \) of the time (necessarily). Given the aforementioned model distribution parametrized by \( p_0 \), and the data set \( \mathcal{D} \), the KL divergence is

\[
D_{\text{KL}} \left( p_{\text{data}}(X) \mid\mid p(X|W) \right) = (1 - f) \log[(1 - f)/p_0] + f \log[f/(1 - p_0)]. \tag{1.34}
\]

Its derivative with respect to \( p_0 \) is

\[
\frac{\partial D_{\text{KL}}}{\partial p_0} = -\frac{(1 - f)}{p_0} + \frac{f}{1 - p_0}. \tag{1.35}
\]

Setting this to zero and solving for \( p_0 \) gives

\[
p_0 = 1 - f, \tag{1.36}
\]

\[
p_1 = f, \tag{1.37}
\]

which is what we would have expected.

**Example 1.1.3** (A visible Boltzmann machine with three units). For an example of a case where an analytic solution is not possible, consider the visible Boltzmann
The energy function looks like

\[ E(x; W) = \sum_{i,j=1}^{3} W_{ij} x_i x_j, \]  

(1.38)

the partition function \( Z(W) \) has the form

\[ Z(W) = \sum_{\text{all } x} e^{-\sum_{i,j=1}^{3} W_{ij} x_i x_j}, \]  

(1.39)

and the log likelihood looks like

\[ \ell(W, D) = -\sum_{x \in D} \sum_{i,j=1}^{3} W_{ij} x_i x_j - N \log Z(W). \]  

(1.40)

Although we can evaluate \( Z(W) \), it has \( 2^3 = 8 \) terms that are nonlinear in the elements of \( W \), and solving the equation \( \partial \ell / \partial W = 0 \) will be impossible to do analytically.

The natural approach to take in this case is to numerically solve for the weights \( W \) that make \( \partial \ell / \partial W \) equal to zero using gradient ascent. To do this we make use of equation (1.23) for the gradient of the likelihood function and iteratively take steps

\[ W_{ij}^{n+1} = W_{ij}^n + \delta W_{ij}^n, \]

where

\[ \delta W_{ij}^n \propto -\left[ \langle \frac{\partial E(x; W^n)}{\partial W_{ij}^n} \rangle_{\text{data}} - \langle \frac{\partial E(x; W^n)}{\partial W_{ij}^n} \rangle_{\text{model}} \right]. \]  

(1.41)

Since \( \partial E(x; W) / \partial W_{ij} = x_i x_j \), this leads to a parameter update rule

\[ \delta W_{ij} \propto - \left[ \langle x_i x_j \rangle_{\text{data}} - \langle x_i x_j \rangle_{\text{model}} \right] = - \left[ \frac{1}{N} \sum_{x \in D} x_i x_j - \sum_{x} x_i x_j \frac{e^{-\sum_{kl} W_{kl} x_k x_l}}{Z(W)} \right]. \]  

(1.42)

Because we can easily evaluate each term in this update rule (due to the important fact that we only need to compute eight terms to evaluate \( Z(W) \)), ML estimation for a three-unit visible Boltzmann machine is tractable.

**Example 1.1.4** (A visible Boltzmann machine with 100 units). What if \( x \) is a 100-dimensional binary vector? In this case, calculating \( Z(W) \) each time we want to compute \( \delta W_{ij} \) for a parameter update is not feasible, as there will be \( 2^{100} \approx 10^{30} \) terms to add together each time. This means that actually evaluating \( p_{\text{model}}(x|W) \) can’t be done exactly, and we can only
approximate it. (The first term in equation 1.23 is perfectly tractable, as it involves an $O(N)$ sum over all points in the dataset $D$.) To approximate the second term in equation 1.23, $\langle \partial E(x; W)/\partial W_{ij} \rangle_{\text{model}}$, we typically sample from the distribution $p_{\text{model}}(x|W)$. In the case of 100-unit binary vectors $x$, having a sampling stage in the inner loop of the gradient ascent algorithm can technically be tractable, but still quite slow on today’s computers. As the number of dimensions increases even more (which is quite a common occurrence, as discussed earlier), the problem as viewed from a maximum likelihood standpoint becomes intractable.

### 1.1.6 When Maximum Likelihood Fails

As we’ve seen, while it is true that for some classes of probabilistic models minimizing the KL divergence between the data and the model distributions is not a difficult task, in the majority of cases one encounters, an analytic solution for the ML estimator $\hat{\theta}_{\text{ML}}$ is both impossible to calculate and intractable to approximate using gradient descent on $D_{\text{KL}}(p_{\text{data}}||p_{\text{model}})$. To see why, consider equation 1.23. The term $\langle \partial E(x; \theta)/\partial \theta \rangle_{\text{data}}$ is usually quite easy to calculate: one computes $E(x; \theta)/N$ for $N$ different values of $x$ and sums them together. The second term, $\langle \partial E(x; \theta)/\partial \theta \rangle_{\text{model}}$, often renders ML learning intractable. As an example, for a $d$-dimensional binary state space, the calculation of the sum over all possible states $x$ involves $2^d$ terms. When $d$ is on the order of 100 or even greater, such as is often the case for spiking neuron data or natural images, the expectation value can only be approximated via sampling techniques. Such techniques usually involve waiting for sampling to converge at each iteration of gradient descent (see Figure 1.5). Having this sampling step in the inner loop of a gradient descent algorithm is what renders learning intractable.

### 1.1.7 Sampling techniques

Many types of sampling techniques are available, among them importance sampling, rejection sampling, Metropolis-Hastings, Gibbs sampling, and hybrid Monte Carlo (HMC) methods [28; 22]. Here we describe the Metropolis method and Gibbs sampling.

#### Sampling using the Metropolis method

The basic idea behind sampling from a distribution is to choose a collection of points in the state space that adequately represent the statistics of that distribution. One typically chooses an initial set of points from a distribution that is trivial to sample from (from a uniform distribution, for example), and then allows each point to evolve...
Figure 1.5: Visualization of estimation of model parameters by sampling. The jagged trajectory represents the evolution of the probability distribution associated with a set of randomly initialized samples under the stochastic sampling dynamics. (a) First learning gradient step. (b) Intermediate learning gradient step. (c) Final learning gradient step.

according to stochastic dynamics with the model distribution as the stationary distribution. Although other sampling techniques exist as well, this type, called Markov chain Monte Carlo (MCMC), is often most successful when the dimensionality of the state space is high. As a very general example of MCMC dynamics, consider the Metropolis method [25, 28]. We wish to draw a sample from a general energy-based model of the form $p(x|\theta) = \exp[-E(x; \theta)]/Z(\theta)$. We first draw a sample $x$ at random (possibly uniformly). We then make use of a proposal distribution to generate a new sample $x'$, given $x$, and will either accept the transition or reject it. Calling the proposal distribution $q(x|x')$ (which of course should also be easy to sample from, and should be symmetric under exchange of $x$ and $x'$), we outline the inner loop of the Metropolis-Hastings procedure as

1. Propose $x'$ by drawing from $q(x'|x)$.
2. If $p(x'|\theta)/p(x|\theta) = \exp[E(x; \theta) - E(x'; \theta)] > 1$, accept the transition.
3. If $\exp[E(x; \theta) - E(x'; \theta)] \leq 1$, accept the transition with probability $\exp[E(x; \theta) - E(x'; \theta)]$.

In any MCMC sampling method, to guarantee that the samples produced are independent and identically distributed, one typically allows for many transitions to become accepted before recording a new sample, as the autocorrelation along the Markov chain dies off as the number of intermediate nodes increases.
Chapter 1. Introduction

Markov chain must be run for significant time (often on the order of 1000 steps) to “burn in” the correct statistics before any samples may be recorded.

Gibbs sampling

Another MCMC sampling method, called Gibbs sampling is quite commonly used for approximating a joint distribution \( p(x_1, \ldots, x_n) \). The idea is to sample each random variable \( x_i \) conditioned on the current observed state of the rest of the random variables. This is especially convenient when the conditional distribution \( p(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \) is simple. The stationary distribution of the Gibbs sampling method is the joint distribution \( p(x_1, \ldots, x_n) \).

Convergence time of MCMC sampling methods

It can be difficult to produce an accurate estimate of how much sampling time must elapse before the samples produced represent the target distribution well. However, to get a ballpark figure, we can look at random-walk type methods like Metropolis-Hastings, where the proposal distribution is just an isotropic Gaussian around the current sample. If we further simplify the sampling dynamics by constraining the radial displacement of each proposed new sample from the current sample to be a constant \( \epsilon \), and any point on the \( \epsilon \)-radius sphere is equally likely to be proposed, we can derive a quick estimate for the sampling time necessary. We assume additionally that we have an acceptance ratio of 100%. After a sampling time \( T \) has elapsed, we will have accumulated \( T \) displacements from our random walk. Labeling the \( t \)-th displacement \( \delta x_t \), we can write the total displacement as

\[
\Delta x = \sum_{t=1}^{T} \delta x_t. \tag{1.43}
\]

We are interested in the length of the vector \( \Delta x \). It makes sense to say that, roughly, sampling will be most of the way to convergence if the length of \( \Delta x \) is comparable to \( \sigma_{\text{max}} \), the standard deviation of the direction in the data with maximum variance.
We can estimate the length of the vector $\Delta x$ as

$$|\Delta x| = \sqrt{\sum_{s=1}^{T} \delta x_s \cdot \sum_{t=1}^{T} \delta x_t} \quad (1.44)$$

$$= \sqrt{\sum_{s=1}^{T} \sum_{t=1}^{T} \delta x_s \cdot \delta x_t}$$

$$= \sqrt{\sum_{s=1}^{T} \sum_{t=1}^{T} \epsilon^2 \cos \theta_{st}}$$

$$= \sqrt{\sum_{t=1}^{T} \epsilon^2 \cos 0 + 2 \sum_{s<t} \epsilon^2 \cos \theta_{st}}$$

$$= \sum_{t=1}^{T} \epsilon^2 + 0 = \epsilon \sqrt{T}, \quad (1.45)$$

Thus, the necessary sampling time is on the order of

$$T \approx \left( \frac{\sigma_{\text{max}}}{\epsilon} \right)^2. \quad (1.46)$$

### 1.2 Alternatives to Maximum Likelihood

Section 1.1.6 has set the stage for a discussion of parameter estimation methods that can get around the roadblock of estimating the partition function. How can you find the best parameters to fit a probabilistic model without actually needing to evaluate the model probabilities themselves?

#### 1.2.1 Contrastive Divergence

When computing a maximum likelihood estimate of the parameters $\theta$, one must often resort to sampling to compute the second term in equation 1.23 for the gradient of the ML objective. This step takes significant time in the inner loop of the estimation computation because of long sampling convergence times. Hinton’s *contrastive divergence* (CD) [14; 42; 9] simply replaces the second term in equation 1.23 with an
estimate of it obtained after very few steps of sampling:

\[ \left\langle \frac{\partial E(x; \theta)}{\partial \theta} \right\rangle_{\text{model}} \to \left\langle \frac{\partial E(x; \theta)}{\partial \theta} \right\rangle_n, \tag{1.47} \]

where \( n \) denotes the number of sampling steps that are allowed to elapse. In Figure\ref{fig:sample_model} this would correspond to stopping the jagged trajectory after only a few kinks. It is common to choose \( n = 1 \) when implementing contrastive divergence. It turns out that following such a gradient minimizes approximately the objective function

\[ J_{\text{CD}} = D_{\text{KL}} (p^{(0)}(x) \| p^{(\infty)}(x|\theta)) - D_{\text{KL}} (p^{(1)}(x) \| p^{(\infty)}(x|\theta)) \] \tag{1.48}

Replacing the model term in the likelihood gradient with a single-step reconstruction not only reduces computational time associated with computing the learning gradient, but also reduces the variance in the gradient caused by sampling noise.

### 1.2.2 Score Matching

Aapo Hyvärinen’s score matching\cite{Hyvaerin1999, Hyvaerin2005, Hyvaerin2005b} provides a conceptually different way around the intractability of the partition function. The objective of score matching is to minimize the square norm of the difference between spatial score functions of the model and data distributions. The spatial score function of a probability density function is defined as

\[ \psi(x; \theta) = \nabla_x \log p(x|\theta). \tag{1.49} \]

Note that, since we can write \( p(x|\theta) = e^{-E(x;\theta)} / Z(\theta) \), we have

\[ \psi(x; \theta) = \nabla_x \left[ \log e^{-E(x;\theta)} - \log Z(\theta) \right] \tag{1.50} \]

or

\[ \psi(x; \theta) = -\nabla_x E(x; \theta). \tag{1.51} \]

Transforming \( p(x|\theta) \) into its score function is a clever way of stripping away the intractability of the partition function. With the score function defined as such, the score matching objective can be defined as the expectation of the square norm of the difference between the data and model score functions:

\[ J_{\text{SM}}(\theta) = \frac{1}{2} \int_{x \in \mathbb{R}^n} \left\| p_{\text{data}}(x) \| \psi(x; \theta) - \psi_{\text{data}}(x) \| \right\|^2 dx, \tag{1.52} \]
Chapter 1. Introduction

and the score matching estimator is defined as

$$
\hat{\theta}_{\text{SM}} = \arg\min_{\theta} J(\theta).
$$

(1.53)

Although the score matching objective doesn’t reference the partition function of the model, it still appears to necessitate the computation of \( \psi_{\text{data}}(x) \), which can be a difficult estimation problem in itself since knowledge of \( p_{\text{data}}(x) \) is typically limited to a finite collection of samples \( D \), and the accuracy of estimating the derivative of a function for which you only know a finite set of samples can be questionable. A way around this problem is to perform integration by parts on equation 1.52. The integrand expands to

$$
J_{\text{SM}}(\theta) = \frac{1}{2} \int_{x \in \mathbb{R}^n} p_{\text{data}}(x) \left( \| \psi_{\text{data}}(x) \|^2 + \| \psi(x; \theta) \|^2 - 2\psi(x; \theta) \cdot \psi_{\text{data}}(x) \right) dx.
$$

(1.54)

The first term in the integrand can be ignored because it is independent of \( \theta \), and the second term has the form

$$
\int_{x \in \mathbb{R}^n} p_{\text{data}}(x) \left( \frac{1}{2} \nabla E \cdot \nabla E \right) dx,
$$

(1.55)

where we take \( \nabla \) to mean a spatial derivative and suppress \( x \) for brevity. The third term has the form

$$
\int_{x \in \mathbb{R}^n} p_{\text{data}}(x) \frac{\nabla E \cdot \nabla p_{\text{data}}}{p_{\text{data}}} dx = \int_{x \in \mathbb{R}^n} \nabla E \cdot \nabla p_{\text{data}} dx.
$$

(1.56)

Replacing \( (\nabla E \cdot \nabla p_{\text{data}}) \) with \( (\nabla \cdot (p_{\text{data}} \nabla E) - p_{\text{data}} \nabla^2 E) \), we have

$$
\int_{x \in \mathbb{R}^n} [\nabla \cdot (p_{\text{data}} \nabla E) - p_{\text{data}} \nabla^2 E] dx,
$$

(1.57)

which can be transformed using the divergence theorem to

$$
\int_{\partial \mathbb{R}^n} p_{\text{data}} \nabla E \cdot dA \hat{n} - \int_{x \in \mathbb{R}^n} p_{\text{data}} \nabla^2 E dx.
$$

(1.58)
Chapter 1. Introduction

Requiring $p_{\text{data}}$ to approach zero at $\infty$ for the sake of normalizability enforces that the first term is zero, and we are left with the transformed objective function

$$J_{\text{SM}}(\theta) = \int_{\mathbf{x} \in \mathbb{R}^n} p_{\text{data}}(\mathbf{x}) \left[ \frac{1}{2} \nabla E \cdot \nabla E - \nabla^2 E \right] d\mathbf{x}. \quad (1.59)$$

The expectation value over the data distribution can be replaced by an average over all points $\mathbf{x} \in \mathcal{D}$ to give

$$J_{\text{SM}}(\theta) = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{D}} \left[ \frac{1}{2} \nabla E \cdot \nabla E - \nabla^2 E \right]. \quad (1.60)$$

Equation 1.60 illustrates that the score matching objective function can be easily computed, insuring that at least gradient descent is a feasible approach to parameter estimation. It is proven in [16] that the score matching objective function leads to a (locally) consistent estimator: if there exists a $\theta$ such that $p_{\text{data}}(\mathbf{x}) = p(\mathbf{x}|\theta)$, then in the limit of large data $\mathcal{D}$, the estimate $\hat{\theta}$ converges in probability to the true parameters $\theta$, as long as the global minimum of $J_{\text{SM}}$ is found. It is also proven [16] that for models $p(\mathbf{x}|\theta)$ in the exponential family, there exists an analytical solution to $\hat{\theta}$. In the case of a multivariate Gaussian, the score matching solution is the same as the maximum likelihood solution.

**Multivariate Gaussian**

The energy function $E$ in the score matching objective has the simple form

$$E(\mathbf{x}; \Sigma, \mu) = \frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu). \quad (1.61)$$

The first and second spatial derivatives are given by

$$\nabla E = \Sigma^{-1} (\mathbf{x} - \mu) \quad (1.62)$$
$$\nabla^2 E = \text{Tr} \Sigma^{-1}, \quad (1.63)$$

and the score matching objective is then

$$J_{\text{SM}} = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{2} (\mathbf{x}_i - \mu)^T \Sigma^{-1} \Sigma^{-1} (\mathbf{x}_i - \mu) - \text{Tr} \Sigma^{-1} \right). \quad (1.64)$$
Chapter 1. Introduction

The equation
\[ \frac{\partial J_{SM}}{\partial \mu_i} = \frac{1}{N} \sum_{i=1}^{N} \Sigma^{-1} \Sigma^{-1} (x_i - \mu) = 0 \] (1.65)
has the solution
\[ \frac{1}{N} \sum_{i=1}^{N} x_i = \mu. \] (1.66)

Using the result
\[ \frac{\partial \Sigma^{-1}_{ij}}{\partial \Sigma_{kl}} = -\Sigma^{-1}_{ik} \Sigma^{-1}_{lj}, \] (1.67)
and the implied result
\[ \frac{\partial \text{Tr} [\Sigma^{-1}]}{\partial \Sigma} = - (\Sigma^{-1})^2, \] (1.68)
we can perform the standard minimax calculation for \( \Sigma \) too (summation over repeated indices is implied):
\[ \frac{\partial J_{SM}}{\partial \Sigma_{rs}} = -\frac{1}{N} \sum_{i=1}^{N} [(x_i - \mu)_j \Sigma^{-1}_{jr} \Sigma^{-1}_{sk} \Sigma^{-1}_{kl} (x_i - \mu)_l - \Sigma^{-1}_{rk} \Sigma^{-1}_{ks}] = 0. \] (1.69)

Right-multiplying by \( \Sigma_{sm} \Sigma_{mn} \Sigma_{rq} \) and summing over repeated indices, we get
\[ -\frac{1}{N} \sum_{i=1}^{N} [(x_i - \mu)_q (x_i - \mu)_n - \Sigma_{qj}] = 0, \] (1.70)
which reduces to the expected result of
\[ \Sigma_{mn} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)_m (x_i - \mu)_n. \] (1.71)

Because we chose an example in the exponential family, score matching allowed us to arrive, with modest effort, at an analytic solution for the best fit parameters. Once the problem space we are considering leaves that happy realm, however, we are forced to consider non-analytic solutions to minimizing the score matching objective.
Chapter 1. Introduction

If we use some flavor of gradient descent, then in addition to computing two spatial derivatives of the energy function, we are also forced to compute another derivative with respect to each model parameter, potentially resulting in a multiple-page-filling expression for the gradient descent update rule.

Score matching has been applied successfully to problems such as learning ICA models [19], and learning XY phase models [8], and has generally made tractable the application of many models that were previously considered intractable due to the inability to compute their partition functions. That said, perhaps one of the more vexing issues with score matching is that it requires that the state space be twice differentiable with respect to spatial coordinates, so applying score matching to discrete state space models is out of the question. In addition, minimizing the score matching objective function using gradient descent requires knowledge of third order derivatives of the model’s energy function, which can often lead to unwieldy, computationally tedious expressions.

Some extensions to score matching may prove useful, such as Hyvärinen’s method of ratio matching for discrete state space models [17] or Lyu’s generalization of score matching [21].

1.2.3 Maximum Pseudolikelihood Learning

Learning by maximum pseudolikelihood, first proposed by Besag [3], is an approach that involves approximating the joint distribution $p(x_1, x_2, \ldots, x_n)$ with a product of all conditional distributions:

$$
p(x_1, x_2, \ldots, x_n) \rightarrow \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \quad (1.72)
$$

$$
:= \prod_{i=1}^{n} p(x_i|x \setminus x_i), \quad (1.73)
$$

where $x \setminus x_i$ denotes the collection of variables $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. As an example, consider an Ising model:

$$
p(x|W) = \exp \left[ - \sum_{ij} W_{ij} x_i x_j \right] Z(W), \quad (1.74)
$$

where the elements of the vector $x$ take on values in $\{0, 1\}$. The joint distribution is difficult to work with because of the intractable partition function $Z(W)$, but the pseudolikelihood function is much easier to compute. The log pseudolikelihood
function has the form

\[ J_{PL} = \log \prod_{i=1}^{n} p(x_i|\mathbf{x} \setminus x_i) \]

\[ = \sum_{i=1}^{n} \log p(x_i|\mathbf{x} \setminus x_i). \]

The conditional probability \( p(x_k|\mathbf{x} \setminus x_k) \) can be computed using Bayes’s rule as

\[ p(x_k|\mathbf{x} \setminus x_k) = \frac{p(\mathbf{x})}{\sum_{x_k} p(\mathbf{x})}, \]

and

\[ \sum_{x_k} p(\mathbf{x}) = \frac{1}{Z(W)} \left[ e^{-\sum_{ij} W_{ij}x_i x_j} + e^{-\sum_{ij} W_{ij}\tilde{x}_i \tilde{x}_j} \right], \]

where \( \tilde{x}_i = x_i \) for \( i \neq k \), and \( \tilde{x}_k = 1 - x_k \). We can expand each sum as

\[ \sum_{ij} W_{ij}x_i x_j = \sum_{i \neq k, j \neq k} W_{ij}x_i x_j + 2 \sum_{j \neq k} W_{kj}x_j x_k + W_{kk}x_k \]

\[ \sum_{ij} W_{ij}\tilde{x}_i \tilde{x}_j = \sum_{i \neq k, j \neq k} W_{ij}x_i x_j + 2 \sum_{j \neq k} W_{kj}(1 - x_k) + W_{kk}(1 - x_k). \]

The conditional probability can then be written

\[ p(x_k|\mathbf{x} \setminus x_k) = \frac{e^{-\sum_{ij} W_{ij}x_i x_j}}{e^{-\sum_{ij} W_{ij}x_i x_j} + e^{-\sum_{ij} W_{ij}\tilde{x}_i \tilde{x}_j}}, \]

which simplifies to

\[ p(x_k|\mathbf{x} \setminus x_k) = \frac{1}{1 + e^{2(2x_k - 1)(\mathbf{w}_k \cdot \mathbf{x} + b_k)}}, \]

where \([\mathbf{w}_k]_i = W_{ki} \) for \( k \neq i \) and \([\mathbf{w}_i]_i = 0 \), and \( b_k = W_{kk} \). The log pseudolikelihood objective function then takes the form

\[ J_{PL} = -\sum_{i=1}^{n} \left[ (2x_i - 1)(\mathbf{w}_i \cdot \mathbf{x} + b_i) + \log \cosh[(2x_i - 1)(\mathbf{w}_i \cdot \mathbf{x} + b_i)] \right]. \]
where we ignore any terms that don’t depend on the parameters. The pseudolikelihood function seems to be throwing away a significant amount of information by considering only the singleton conditional densities, but it is in fact the case that the joint distribution of random variables is completely determined by the ensemble of conditional distributions (Brook’s Lemma [6]). The result is a very useful and fast estimator that has been found to be consistent in several cases.

1.2.4 Minimum Velocity Learning

Javier Movellan has developed an approach called minimum velocity learning [26] for diffusion networks [27]. A diffusion network consists of $n$ continuous-valued nodes $X_t$ whose stochastic activation dynamics, on average, diffuse along the gradient of a potential function $E(X_t)$:

$$dX_t = -\nabla E(X_t)dt + \sqrt{2}dB_t,$$  

(1.84)

where $E(x)$ is the potential associated with the activation state $x$, and $dB_t$ is a noise term. The activation dynamics described by equation 1.84 correspond to the evolution of a probability distribution $p_t(x)$ via Fokker-Planck-Kolmogorov dynamics that conserve probability:

$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot J_t(x)$$ \hspace{1cm} (1.85)

$$J_t(x) := p_t(x)V_t(x)$$ \hspace{1cm} (1.86)

$$V_t(x) := \nabla \left( -E(x) - \log p_t(x) \right),$$ \hspace{1cm} (1.87)

where $p_t(x)$ describes the probability distribution of activations $x$ at time $t$, $J_t$ is a probability current, and $V_t$ is a probability velocity. Movellan’s approach is to choose parameters of the model that minimize the expected square norm of the probability velocity $V_t$ at $t = 0$ when the network $X_0$ is initialized to the data:

$$\min \mathbb{E} \left[ \| V_0(x) \|^2 \right].$$ \hspace{1cm} (1.88)

It turns out that this minimum velocity approach is equivalent to both Hyvärinen’s method of score matching and Hinton’s single-step constrastive divergence when the state space of the model is continuous. The objective function for minimum velocity
learning can be calculated to be
\[
\mathbb{E} \left[ \left\| \mathbf{V}_0 \right\|^2 \right] = \int p_0(x) \left[ \nabla E \cdot \nabla E - 2 \nabla E \cdot \nabla \log p_0(x) \right] \, dx
\]
\[
= \int \left[ p_0(x) \nabla E(x) \cdot \nabla E(x) - 2 \nabla E(x) \cdot \nabla p_0(x) \right] \, dx
\]
\[
= \int p_0(x) \left[ \nabla E(x) \cdot \nabla E(x) - 2 \nabla^2 E(x) \right] \, dx
\approx \frac{2}{N} \sum_{x \in D} \left[ \frac{1}{2} \nabla E(x) \cdot \nabla E(x) - \nabla^2 E(x) \right],
\]
(1.89)
which is the same as the score matching objective up to a factor of two.

To see the connection to contrastive divergence, we first observe that the diffusion network dynamics defined by equation 1.84 effectively define a sampling process that converges to the model distribution defined by \( E(x) \). Letting \( t \) steps of this sampling process elapse brings us to the state \( X_t \) in the sampling Markov chain. The approximate objective function for CD-\( t \) looks like
\[
J_{CD_t} = D_{KL} \left( p_0(x) \left| \left| p_{\infty}(x) \right| \right| \right) - D_{KL} \left( p_t(x) \left| \left| p_{\infty}(x) \right| \right| \right),
\]
which can be expanded to
\[
\left[ \langle E(x) \rangle_{p_0} - H(p_0) \right] - \left[ \langle E(x) \rangle_{p_t} - H(p_t) \right],
\]
(1.91)
where \( H(p) = -\sum_x p(x) \log p(x) \) is the Shannon entropy of the distribution \( p \). Note that the terms \( [\langle E(x) \rangle - H(p)] \) are precisely the Helmholtz free energies \( F \) of their corresponding distributions, with Boltzmann’s constant and the temperature \( T \) set to 1, so we can write the approximate contrastive divergence objective as
\[
J_{CD_t} = F_0 - F_t.
\]
(1.92)
As \( t \) approaches 0, or approximately equivalently, as the \( n \) in CD\( n \) approaches 1 (single-step CD), the approximate objective approaches
\[
J_{CD_1} = \frac{F_0 - F_1}{1} \approx - \frac{\partial F_t}{\partial t} \bigg|_{t=0}.
\]
(1.93)
We can see from equations 1.90 and 1.91 that
\[
\frac{\partial D_{KL} \left( p_t(x) \left| \left| p_{\infty}(x) \right| \right| \right)}{\partial t} = \frac{\partial F_t}{\partial t}.
\]
(1.94)
Chapter 1. Introduction

Omitting spatial arguments for brevity, we can see that

\[
\frac{\partial D_{KL}(p_t \| p_\infty)}{\partial t} = \frac{\partial}{\partial t} \int dx \left[ p_t \log p_t - p_t \log p_\infty \right] = \int dx \left[ \frac{\partial p_t}{\partial t} (\log p_t - \log p_\infty + 1) \right] = \int dx \left[ -\nabla \cdot (p_t \mathbf{V}_t) (\log p_t - \log p_\infty + 1) \right],
\]

(1.95)

where we used equations 1.85 to expand the time derivative of \( p_t \). Applying the product rule, we can write this as

\[
\frac{\partial D_{KL}(p_t \| p_\infty)}{\partial t} = -\int dx \left\{ \nabla \cdot [p_t \mathbf{V}_t (\log p_t - \log p_\infty + 1)] - p_t \mathbf{V}_t \cdot \nabla (\log p_t - \log p_\infty + 1) \right\} = -\int_{\mathbb{R}^n} d\Sigma \cdot [p_t \mathbf{V}_t (\log p_t - \log p_\infty + 1)] + \int dx p_t \mathbf{V}_t \nabla (\log p_t - \log p_\infty + 1)
\]

\[
= \int dx p_t \mathbf{V}_t \nabla (\log p_t - \log p_\infty + 1)
\]

\[
= \int dx p_t \mathbf{V}_t \cdot \nabla (E + \log p_t)
\]

\[
= -\int dx p_t \mathbf{V}_t \cdot \mathbf{V}_t
\]

\[
= -\mathbb{E} \left[ \| \mathbf{V}_t \|^2 \right],
\]

(1.103)

where the boundary integral must evaluate to zero in order for the probability density \( p_t \) to be normalizable. Thus, we have the result that

\[
J_{CD_1} \approx -\left. \frac{\partial F_t}{\partial t} \right|_{t=0} = \mathbb{E} \left[ \| \mathbf{V}_0 \|^2 \right],
\]

(1.104)

which immediately implies that the approximate contrastive divergence objective approaches that of minimum velocity learning and also that of score matching as the number of sampling steps goes to one.

Movellan’s minimum velocity learning is exciting because in addition to providing the justification for learning the parameters of a diffusion network using score matching, it also unifies the ideas behind score matching and contrastive divergence under a
single framework by lifting from the stochastic sampling dynamics behind contrastive divergence to the deterministic diffusion dynamics of the distribution over samples. There still remains, however, the limitation that the network states $X_t$ must live in a continuous space so that their spatial derivatives exist.
Chapter 2

Minimum Probability Flow Learning

Although finding methods of evaluating $Z(\theta)$ is at the time of writing an active field of research, and there exist various methods for approximating it (among them Neal’s annealed importance sampling and other Monte Carlo methods \[28, 22\]), the central result of this report renders the partition function irrelevant to learning model parameters. As is discussed in section 1.2, other methods exist for learning in cases for which evaluating or even approximating $Z(\theta)$ is not an option. However, each method has its own shortcomings: lack of objective function, restriction to continuous state spaces, unwieldy calculations, restricted parameter domains, \textit{et cetera}.

The central idea behind Minimum Probability Flow (MPF) learning takes part of its inspiration from Hinton’s contrastive divergence: instead of traveling as far down the Markov chain as it takes to produce true samples from the model distribution, only take a few steps towards evaluating the model term in the log likelihood gradient. Contrastive divergence relies on the fact that even though you are only taking a small number of sampling steps during each learning iteration, those steps always tend to take you closer to the model distribution. We will see that one of the crucial differences between MPF learning and contrastive divergence is that MPF utilizes deterministic linear dynamics instead of stochastic dynamics.

As we will see, and as has been hinted at previously, this central idea is actually at the core of each of contrastive divergence, score matching, and minimum velocity learning. Minimum probability flow builds off of that central idea as its core tenet, and provides a flexible and powerful viewpoint from which we can approach general parametric density estimation problems that each of the previously described methods only partially addresses.

We will approach this narrative by addressing the following points:

- Partition function intractability motivates a class of learning methods that mini-
mize diffusive flow of probability from the data distribution to a candidate model distribution as the learning objective, rather than maximizing log likelihood.

- We can view linear diffusive flow as a special case of a general state space picture of diffusion.

- With appropriate boundary conditions and a few loose constraints applied, this general state space picture can be used to create a general framework for learning by minimization of diffusive flow of probability across state space. We call this framework Minimum Probability Flow Learning.

- This general learning model can be directly and efficiently applied to a diverse range of estimation problems.

- We derive the previously-discussed prior art as a limiting case of minimum probability flow.

2.1 Saying Goodbye to the Partition Function

The reason for the headache associated with ML learning is the second term on the right hand side of equation 1.23 (which I will call the model term), reproduced here for brevity:

\[-\frac{\partial}{\partial \theta} D_{\text{KL}}(p_{\text{data}}(x) \| p_X(x|\theta)) = -N \left[ \left\langle \frac{\partial E(x; \theta)}{\partial \theta} \right\rangle_{\text{data}} - \left\langle \frac{\partial E(x; \theta)}{\partial \theta} \right\rangle_{\text{model}} \right].\]

When the dimensionality of the state space becomes too high, which is more often than not the case for datasets of interest, the model term presents us with difficulty. We are going to assume here that evaluating or approximating the partition function is out of the question. As discussed earlier, an approach that has seen increasingly wide adoption in the community is Aapo Hyvärinen’s score matching method. This method attempts to minimize the difference between score functions of the data and model distributions, and these score functions manage to disentangle and remove the partition function from the objective function by applying spatial derivatives to the log probability distributions. In 2008, Javier Movellan released a manuscript in which he recast the objective of several learning methods, score matching among them, in terms of the minimization of a probability velocity under Gaussian diffusion. The main gist of this work was that one could use a form of diffusion dynamics to evolve any distribution of probability over a continuous state space into the model distribution, and then choose as your objective the minimization of the initial velocity of this evolution.
Chapter 2. Minimum Probability Flow Learning

Any problem over a continuous state space that can be approached using score matching can be seen from Movellan’s standpoint as minimizing diffusive flow. A natural question one might ask is whether this idea can be generalized and extended to any discrete space as well. The answer is yes, which is one of the main points of this research, but to give a more nuanced answer, it is helpful to think about diffusion over a continuous state space. Understanding diffusion as a time-evolved application of a general transfer matrix to the initial configuration is our first stepping stone.

2.2 Prelude: Lifting Linear Diffusion Processes to a General State Space Picture

Continuous state space diffusion \cite{29} can be described by a Fokker-Planck equation of the form

\[
\frac{\partial f(x,t)}{\partial t} = B \nabla \cdot (\nabla V(x)f(x,t)) + B k_B T \nabla^2 f(x,t), \tag{2.1}
\]

where \(f(x,t)\) is the fluid density at a point in space \(x\) and time \(t\), \(B\) is called the mobility, \(T\) is the fluid’s temperature, \(k_B\) is Boltzmann’s constant, and \(V(x)\) is a spatially varying potential field. Can we write down a discrete state space analog to this equation? To answer this question, we will first recast the dynamics into a more general form.

The diffusion process described by equation 2.1 involves the time evolution of a fluid density evaluated at every point in space \(x\). As time progresses, fluid can move from one point in space to “nearby” points in space via small, local flows. Now suppose that we can factor out the time dependence of the local flow rates between points in space (denoted as \(R(x,x',t)\)) as follows:

\[
R(x,x',t) = \Gamma(x,x')f(x',t), \tag{2.2}
\]

where \(\Gamma(x,x')\) is a time- and density-independent transition rate. Here, by \(R(x,x',t)\) we mean the rate of fluid flow out of the point \(x'\) and into the point \(x\). Then, the total flow rate into the point \(x\) from all other points \(x'\) can be written as

\[
R_{in}(x,t) = \int \Gamma(x,x')f(x',t)d\!x'. \tag{2.3}
\]

Similarly, the total flow rate out of the point \(x\) and into all other points \(x'\) can be
written as

\[ R_{\text{out}}(x, t) = \int \Gamma(x', x) f(x, t) dx'. \tag{2.4} \]

What cannot occur is for the total amount of fluid to change: there is no mechanism in equation 2.1 for matter to be created or annihilated. For this reason, the rate of change in fluid density \( f(x, t) \) must be accounted for by flow into \( x \) from all other states \( x' \) and flow out of \( x \) into all other states \( x' \), as expressed by the master equation:

\[
\frac{\partial f(x, t)}{\partial t} = R_{\text{in}}(x, t) - R_{\text{out}}(x, t)
= \int [\Gamma(x, x') f(x', t) - \Gamma(x', x) f(x, t)] dx'. \tag{2.5}
\]

Equation 2.5 is very general, but under certain conditions it actually reduces to equation 2.1. To see this, we assume that the transition rate \( \Gamma(x', x) \) is highly peaked about \( x \) if we hold \( x' \) fixed. This will allow us to Taylor expand equation 2.5 in the width of the peak in \( \Gamma \).

First define

\[ \Gamma(x; \Delta x) = \Gamma(x + \Delta x, x), \tag{2.6} \]

so that \( \Gamma(x; \Delta x) \) is the transition rate from the state \( x \) to the state \( x + \Delta x \). The master equation 2.5 can then be written

\[
\frac{\partial f(x, t)}{\partial t} = \int \left\{ f(x + \Delta x) \Gamma(x + \Delta x; -\Delta x) - f(x, t) \Gamma(x; \Delta x) \right\} d\Delta x. \tag{2.7}
\]

We can then Taylor expand \( f(x + \Delta x) \) and \( \Gamma(x + \Delta x; -\Delta x) \) to second order in \( \Delta x \):

\[
f(x + \Delta x) \approx f(x, t) + \Delta x \cdot \nabla f(x, t) + \frac{1}{2} \sum_{ij} \Delta x_i \Delta x_j \nabla_i \nabla_j f(x, t) \tag{2.8}
\]

\[
\Gamma(x + \Delta x; -\Delta x) \approx \Gamma(x; -\Delta x) + \Delta x \cdot \nabla \Gamma(x; -\Delta x) + \frac{1}{2} \sum_{ij} \Delta x_i \Delta x_j \nabla_i \nabla_j \Gamma(x; -\Delta x) \tag{2.9}
\]
Then, to second order in $\Delta x$ we can write
\[
f(x + \Delta x)\Gamma(x + \Delta x; -\Delta x) \approx f(x, t)\Gamma(x; -\Delta x) \\
+ \Delta x \cdot \nabla(f(x, t)\Gamma(x; -\Delta x)) \\
+ \frac{1}{2} \sum_{ij} \Delta x_i \Delta x_j \nabla_i \nabla_j (f(x, t)\Gamma(x; -\Delta x)).
\] (2.10)

The master equation (equation 2.5) then reads
\[
\frac{\partial f(x, t)}{\partial t} \approx \int \{ f(x, t)\Gamma(x; -\Delta x) - f(x, t)\Gamma(x; \Delta x) \\
+ \Delta x \cdot \nabla(f(x, t)\Gamma(x; -\Delta x)) \\
+ \frac{1}{2} \sum_{ij} \Delta x_i \Delta x_j \nabla_i \nabla_j (f(x, t)\Gamma(x; -\Delta x))\} \, d\Delta x.
\] (2.11)

Since $\Delta x$ is a dummy variable in the integral, we can substitute $\Delta x \to -\Delta x$ in terms with $\Gamma(x; -\Delta x)$ to get rid of the negative sign. This causes the first and second terms to cancel, and the master equation then reads
\[
\frac{\partial f(x, t)}{\partial t} \approx -\nabla \cdot \left\{ f(x, t) \int \Delta x \, \Gamma(x; \Delta x) \, d\Delta x \right\} \\
+ \frac{1}{2} \sum_{ij} \nabla_i \nabla_j \left\{ f(x, t) \int \Delta x_i \Delta x_j \, \Gamma(x; \Delta x) \, d\Delta x \right\}.
\] (2.12)

The first integral is the expected displacement rate:
\[
v(x, t) \equiv \langle \Delta x \rangle_{\Delta t}. \tag{2.13}
\]

The quantity $\Gamma(x; \Delta x)\Delta t$ is the probability that a fluid “particle” will transition from the point $x$ to the point $x + \Delta x$ within a time $\Delta t$. One can think of this displacement rate as the expected bulk velocity of the fluid. The second integral is the covariance of the displacement. We assume that this takes the form
\[
\langle \Delta x_i \Delta x_j \rangle_{\Delta t} = \delta_{ij} \langle \Delta x_i^2 \rangle_{\Delta t}, \tag{2.14}
\]
which allows us to write the master equation as
\[
\frac{\partial f(x, t)}{\partial t} = -\nabla \cdot \{ f(x, t)v(x, t) \} + \frac{1}{2} \sum_i \nabla_i^2 \left\{ f(x, t) \frac{\langle \Delta x_i^2 \rangle_{\Delta t}}{\Delta t} \right\}.
\] (2.15)
Chapter 2. Minimum Probability Flow Learning

The quantities \( v(x, t) \) and \( \langle \Delta x_i^2 \rangle_{\Delta t} / \Delta t \) can be determined from the equation

\[
\frac{dv}{dt} = -\frac{v}{MB} + \frac{F(t)}{M}, \tag{2.16}
\]

where the term \( F(t)/M \) represents a random disturbing force with zero mean that is uncorrelated in time: \( \langle F(t) \rangle_t = 0 \). Using the identities

\[
\begin{align*}
\mathbf{r} \cdot \frac{dv}{dt} &= \frac{d}{dt} (\mathbf{r} \cdot \mathbf{v}) - v^2 \tag{2.17} \\
\frac{d}{dt} \mathbf{r}^2 &= 2 \mathbf{r} \cdot \mathbf{v}, \tag{2.18}
\end{align*}
\]

and the fact that

\[
\langle \mathbf{r} \cdot F(t) \rangle_t = 0, \tag{2.19}
\]

we can write this equation in the form

\[
\frac{d^2}{dt^2} \langle r^2 \rangle_t + \frac{1}{MB} \frac{d}{dt} \langle r^2 \rangle_t = 2 \langle v^2 \rangle_t. \tag{2.20}
\]

At quasistatic equilibrium, the equipartition theorem states that \( \langle v^2 \rangle_t = dk_B T/M \), where \( d \) is the number of spatial dimensions. This equation then has the solution

\[
\langle r^2 \rangle_t = 2dk_B T \left[ t - MB (1 - \exp (-t/MB)) \right], \tag{2.21}
\]

with \( \langle r^2 \rangle_t (0) = 0 \) and \( d \langle r^2 \rangle_t (0)/dt = 0 \). For times that are long compared with \( MB \), we have

\[
\langle r^2 \rangle_t = (2dBk_B T)t. \tag{2.22}
\]

By symmetry, this implies that

\[
\langle \Delta x_i^2 \rangle_t = (2Bk_B T)t. \tag{2.23}
\]

We can then conclude that

\[
\frac{\langle \Delta x_i^2 \rangle_{\Delta t}}{\Delta t} = 2Bk_B T. \tag{2.24}
\]

At times \( \Delta t \) long compared with \( MB \) but short compared with the time scales in the dynamics imposed by the potential \( V(x) \), fluid particles have reached an effective terminal velocity determined by equation [2.16]. At such time scales, we can approximate
the velocity $v(x, t)$ as

$$- \frac{v}{B} - \nabla V(x) = 0, \quad (2.25)$$

or

$$v = -B \nabla V(x). \quad (2.26)$$

This allows us to write equation 2.15 as

$$\frac{\partial f(x, t)}{\partial t} = B \nabla \cdot \{ \nabla V(x)f(x, t) \} + Bk_B T \nabla^2 f(x, t). \quad (2.27)$$

This shows that the diffusion equation 2.1 is a special case of the more general master equation defined by transition rates $\Gamma$, in the limit that transition probabilities are sharply peaked around initial states.

**Matrix form of the master equation**

This master equation (equation 2.5) can be expressed in the form

$$\frac{\partial f(x, t)}{\partial t} = \Gamma \{ f(x, t) \}, \quad (2.28)$$

where the linear operator $\Gamma \{ \}$ operates on $f$ as

$$\Gamma \{ f \}(x, t) = \int [\Gamma(x, x')f(x', t) - \Gamma(x', x)f(x, t)] \, dx'. \quad (2.29)$$

Now that we are treating the linear diffusion dynamics in a more general form, we are at liberty to generalize the state space as well. The linear operator form, in particular, suggests that we can write down a discrete version of the diffusion dynamics

$$\frac{\partial f_i(t)}{\partial t} = \sum_j \Gamma_{ij} f_j(t) - \sum_j \Gamma_{ji} f_i(t), \quad (2.30)$$

where the state space is now indexed by a discrete index $i$ rather than by a continuous index $x$, and $\Gamma_{ij}$ represents the transition rate from the state $j$ to the state $i$. If we define the diagonal elements of $\Gamma$ such that

$$\Gamma_{ii} = - \sum_j \Gamma_{ji} \quad (j \neq i), \quad (2.31)$$
then the master equation 2.30 takes the compact matrix form
\[
\frac{\partial f(t)}{\partial t} = \Gamma f(t).
\]  
(2.32)

This ability to cast the diffusion dynamics into both continuous and discrete forms came from lifting away from the specific case of Gaussian diffusion. This abstraction exposes the inner workings of the state transitions behind those dynamics through the linear operator \( \Gamma \). It is this liberation from the bondage of diffusion, in combination with a set of guidelines for choosing the structure of \( \Gamma \), that is at the heart of minimum probability flow learning.

2.3 The Minimum Probability Flow Learning Algorithm

We are now in a position to leverage our generalized diffusion dynamics towards a powerful non-maximum-likelihood learning method in the spirit of contrastive divergence, score matching, and minimum velocity learning, but defined over arbitrary state spaces: minimum probability flow learning.

We will use the concepts of conservation of probability and detailed balance to adapt the previously-discussed generalized diffusion process to define dynamics on the space of probability distributions that converge to a model distribution. This process will then be used to generalize the concepts behind score matching, CD and minimum velocity learning to arbitrary state spaces.

Our goal is to find the parameters that cause a probabilistic model to best agree with a list \( D \) 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 \( \epsilon \) (see Figure 2.1).

2.3.1 Distributions

The data distribution is represented by a vector \( p^{(0)} \), with \( p_i^{(0)} \) the fraction of the observations \( D \) in state \( i \). The superscript \( (0) \) represents time \( t = 0 \) under the system dynamics (which will be described in more detail in Section 2.3.2). For example, in a two variable binary system, \( p^{(0)} \) would have four entries representing the fraction of the data in states 00, 01, 10 and 11 (Figure 2.2).

Our goal is to find the parameters \( \theta \) that cause a model distribution \( p^{(\infty)}(\theta) \) to best match the data distribution \( p^{(0)} \). The superscript \( (\infty) \) on the model distribution
Chapter 2. Minimum Probability Flow Learning

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

**Progression of Learning**

- data distribution
- model distribution
- \(\bullet\) slightly transformed distribution
- \(--\) model parameters
- \(\Rightarrow\) dynamical trajectory
indicates that this is the equilibrium distribution reached after running the dynamics for infinite time. Without loss of generality, we assume the model distribution is of the form

$$p_i(\infty) = \frac{\exp(-E_i(\theta))}{Z(\theta)} = \exp(-E_i(\theta)) Z(\theta),$$

(2.33)

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

$$Z(\theta) = \sum_i \exp(-E_i(\theta))$$

(2.34)

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

### 2.3.2 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 $p^{(t)}$ [29]:

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

(2.35)

where $\dot{p}^{(t)}_i$ is the time derivative of $p^{(t)}_i$. 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 (2.35) 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 $\theta$ results from the requirement that the chosen dynamics cause $p^{(t)}$ to flow to the equilibrium distribution $p^{(\infty)}(\theta)$. For readability, explicit dependence on $\theta$ will be dropped except where necessary. If we choose to set the diagonal elements of $\Gamma$ to obey $\Gamma_{ii} = -\sum_{j \neq i} \Gamma_{ji}$, then we can write the dynamics as

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

(2.36)

(see figures [2.2 and 2.3]). The unique solution for $p^{(t)}$ is given by

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

(2.37)

where $\exp(\Gamma t)$ is a matrix exponential. It is important to note that the form chosen for $\Gamma$ in equation (2.36), coupled with the satisfaction of detailed balance and ergodicity
Figure 2.2: Dynamics of minimum probability flow learning. Model dynamics represented by the probability flow matrix $\Gamma$ (*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.
Figure 2.3: Another representation of the dynamics generated by equation 2.35.
introduced in section 2.3.3 guarantees that there is a unique eigenvector \( p^{(\infty)} \) of \( \Gamma \) with eigenvalue zero, and that all other eigenvalues of \( \Gamma \) have negative real parts. This implies a flow field of the form shown in figure 2.4.

### 2.3.3 Detailed Balance

The second core concept is detailed balance,

\[
\Gamma_{ji} p_i^{(\infty)}(\theta) = \Gamma_{ij} p_j^{(\infty)}(\theta),
\]

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 \( p^{(\infty)}(\theta) \) is a fixed point of the dynamics. Note that detailed balance is a sufficient but not necessary condition for a fixed point as \( t \to \infty \). One can achieve a fixed point with cycles of probability \( i \to j \to k \to \cdots \to i \) as well. For the development of minimum probability flow, we will assume that (strong) detailed balance holds. Sampling in most Monte Carlo methods is performed by choosing \( \Gamma \) consistent with equation 2.38 (and the added requirement of ergodicity), then stochastically running the dynamics of equation 2.35. Note that there is no need to restrict the dynamics defined by \( \Gamma \) to those of any real physical process, such as diffusion.

Equation 2.38 can be written in terms of the model’s energy function \( E(\theta) \) by substituting in equation 2.33 for \( p^{(\infty)}(\theta) \):

\[
\Gamma_{ji} \exp(-E_i(\theta)) = \Gamma_{ij} \exp(-E_j(\theta)).
\]

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

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

where the connectivity function

\[
g_{ij} = g_{ji} = \begin{cases} 
0 & \text{unconnected states} \\
1 & \text{connected states} 
\end{cases} \quad (i \neq j)
\]

determines which states are allowed to directly exchange probability with each other\(^1\).

\(^1\)The non-zero \( \Gamma \) may also be sampled from a proposal distribution rather than set via a deter-
Figure 2.4: A flow field representation of the dynamics generated by equations 2.35 and 2.38. There exists a unique eigenvector $p^{(\infty)}$ of $\Gamma$ with eigenvalue zero, and the remaining eigenvalues have negative real parts. These conditions guarantee exponential decay of transients to an equilibrium distribution.
Chapter 2. Minimum Probability Flow Learning

The coefficients $g_{ij}$ can be set such that $\mathbf{\Gamma}$ is extremely sparse (see Section 2.3.5). 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).

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

$$\hat{\theta}_{\text{ML}} = \arg\min_{\theta} D_{KL}(p^{(0)} || p^{(\infty)}(\theta)) \tag{2.42}$$

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

$$\hat{\theta}_{\text{MPF}} = \arg\min_{\theta} K(\theta) \tag{2.43}$$

$$K(\theta) = D_{KL}(p^{(0)} || p^{(\epsilon)}(\theta)) \tag{2.44}$$

ministic scheme, in which case $g_{ij}$ takes on the role of proposal distribution — see section 2.3.8.
For small $\epsilon$, $D_{KL}(p^{(0)}||p^{(\epsilon)}(\theta))$ can be approximated by a first order Taylor expansion, 

\[
K(\theta) \approx D_{KL}(p^{(0)}||p^{(t)}(\theta)) \bigg|_{t=0} + \epsilon \frac{\partial D_{KL}(p^{(0)}||p^{(t)}(\theta))}{\partial t} \bigg|_{t=0}
\]

\[
= 0 + \epsilon \frac{\partial D_{KL}(p^{(0)}||p^{(t)}(\theta))}{\partial t} \bigg|_{t=0}
\]

\[
= \epsilon \frac{\partial}{\partial t} \left( \sum_{i \in D} p_i^{(0)} \log \frac{p_i^{(0)}}{p_i^{(t)}} \right) \bigg|_{0}
\]

\[
= -\epsilon \sum_{i \in D} \frac{\partial p_i^{(0)}}{\partial t} \bigg|_{0}
\]

\[
= -\epsilon \left( \frac{\partial}{\partial t} \sum_{i \in D} p_i^{(t)} \right) \bigg|_{0}
\]

\[
= -\epsilon \left( \frac{\partial}{\partial t} \left( 1 - \sum_{i \neq D} p_i^{(t)} \right) \right) \bigg|_{0}
\]

\[
= \epsilon \sum_{i \neq D} \frac{\partial p_i^{(t)}}{\partial t} \bigg|_{0}
\]

\[
= \epsilon \sum_{i \neq D} \sum_{j \in D} \Gamma_{ij} p_j^{(0)}
\]

\[
= \epsilon \frac{1}{|D|} \sum_{j \in D} \sum_{i \neq D} \Gamma_{ij},
\]

where we used the fact that $\sum_{i \in D} p_i^{(t)} + \sum_{i \neq D} p_i^{(t)} = 1$ in equation 2.51. Thus we see that $K(\theta)$ is a measure of the flow of probability, at time $t = 0$ under the dynamics, out of data states $j \in D$ into non-data states $i \notin D$,

\[
K(\theta) = \frac{\epsilon}{|D|} \sum_{j \in D} \sum_{i \neq D} g_{ij} \exp \left[ \frac{1}{2} \left( E_j(\theta) - E_i(\theta) \right) \right],
\]
Chapter 2. Minimum Probability Flow Learning

with gradient

\[
\frac{\partial K(\theta)}{\partial \theta} = \frac{\epsilon}{|D|} \sum_{j \in D} \sum_{i \neq D} \left[ \frac{\partial E_j(\theta)}{\partial \theta} - \frac{\partial E_i(\theta)}{\partial \theta} \right] g_{ij} \exp \left[ \frac{1}{2} (E_j(\theta) - E_i(\theta)) \right],
\]  

(2.56)

where $|D|$ is the number of observed data points. Note that equations (2.54) and (2.56) do not depend on the partition function $Z(\theta)$ or its derivatives.

$K(\theta)$ is uniquely zero when $p(0)$ and $p(\infty)(\theta)$ are equal. This implies consistency, in that if the data comes from the model class, in the limit of infinite data $K(\theta)$ will be minimized by exactly the right $\theta$. In addition, $K(\theta)$ is convex for all models $p(\infty)(\theta)$ in the exponential family — that is, models whose energy functions $E(\theta)$ are linear in their parameters $\theta$ [23] (see section 2.5).

### 2.3.5 Tractability

The dimensionality of the vector $p(0)$ is typically huge, as is that of $\Gamma$ (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 $\Gamma_{ij}$ corresponding to data states, $j \in D$. 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 $O(|D|)$, and does not depend on the number of system states, only on the (much smaller) number of observed data points.

### 2.3.6 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. $\Gamma_{ji}, g_{ji}$, and $p^{(t)}_{i}(x)$ become continuous functions $\Gamma(x_j, x_i), g(x_j, x_i)$, and $p^{(t)}_{i}(x)$. $\Gamma(x_j, x_i)$ can be populated stochastically and extremely sparsely (see section 2.3.8), preserving the $O(|D|)$ cost. A specific scheme (similar to CD with Hamiltonian Monte Carlo) for estimating parameters in a continuous state space via MPF is described in section 2.3.9.

### 2.3.7 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 (e.g. 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.

In continuous state spaces, the estimated parameters are much more sensitive to the choice of \( g(x_j, x_i) \). One effective form for \( g(x_j, x_i) \) is described in the next sections, but theory supporting different choices of \( g(x_j, x_i) \) remains an area of active exploration. The content of sections 2.3.8 and 2.3.9 are included from [35] and [37].

2.3.8 Sampling the Connectivity Matrix \( \Gamma \)

The MPF learning scheme is blind to regions in state space which data states don’t have any connectivity to — the flow at time 0 is only a function of the states that are directly connected to data states. To get the most informative learning signal, it seems sensible to encourage probability flow directly between data states and states that are probable under the model. That way the objective function is sensitive to the regions which are probable under the model. We believe nearest neighbor connectivity schemes are effective largely because, as the parameters converge, the regions around data states become the high probability regions for the model. We wish to try connectivity schemes other than nearest neighbors to allow probability to most efficiently flow between data states and high probability model states. In order to do so we need to slightly extend the MPF algorithm. We do this by allowing the connectivity pattern in \( \Gamma \) to be sampled independently in every infinitesimal time step.

Since \( \Gamma \) is now sampled, we will modify detailed balance to demand that, averaging over the choices for \( \Gamma \), the net flow between pairs of states is 0.

\[
\langle \Gamma_{ji} \rangle p_i^{(\infty)}(\theta) = \langle \Gamma_{ij} \rangle p_j^{(\infty)}(\theta), \quad (2.57)
\]

where the ensemble average is over the connectivity scheme for \( \Gamma \). We describe the connectivity scheme via a function \( g_{ij} \), such that the probability of there being a connection from state \( j \) to state \( i \) at any given moment is \( g_{ij} \). We also introduce a function \( F_{ij} \), which provides the value \( \Gamma_{ij} \) takes on when a connection occurs from \( j \) to \( i \). That is, it is the probability flow rate when flow occurs:

\[
\langle \Gamma_{ij} \rangle = g_{ij} F_{ij}. \quad (2.59)
\]

Detailed balance now becomes

\[
g_{ji} F_{ji} p_i^{(\infty)}(\theta) = g_{ij} F_{ij} p_j^{(\infty)}(\theta). \quad (2.60)
\]
Solving for $\mathbf{F}$ we find

$$
\frac{F_{ij}}{F_{ji}} = \frac{g_{ji}}{g_{ij}} \frac{p_i^{(\infty)}(\theta)}{p_j^{(\infty)}(\theta)} = \frac{g_{ji}}{g_{ij}} \exp \left[ E_j(\theta) - E_i(\theta) \right].
$$

(2.61)

$\mathbf{F}$ is underconstrained by the above equation. Motivated by symmetry and aesthetics, we choose as the form for the (non-zero, non-diagonal) entries in $\mathbf{F}$

$$
F_{ij} = \left( \frac{g_{ji}}{g_{ij}} \right)^{\frac{1}{2}} \exp \left[ \frac{1}{2} (E_j(\theta) - E_i(\theta)) \right].
$$

(2.62)

$\Gamma$ is now populated as

$$
\Gamma_{ij} = \begin{cases} 
- \sum_{k \neq i} \Gamma_{ki} & \text{if } i = j \\
F_{ij} & \text{if } r_{ij} < g_{ij} \text{ and } i \neq j \\
0 & \text{if } r_{ij} \geq g_{ij} \text{ and } i \neq j
\end{cases}
$$

(2.63)

Similarly, its average value can be written as

$$
\langle \Gamma_{ij} \rangle = g_{ij} \left( \frac{g_{ji}}{g_{ij}} \right)^{\frac{1}{2}} \exp \left[ \frac{1}{2} (E_j(\theta) - E_i(\theta)) \right]
$$

(2.65)

$$
= (g_{ij} g_{ji})^{\frac{1}{2}} \exp \left[ \frac{1}{2} (E_j(\theta) - E_i(\theta)) \right].
$$

(2.66)

So, we can use any connectivity scheme $\mathbf{g}$ in learning. We just need to scale the non-zero, non-diagonal entries in $\Gamma$ by $\left( \frac{g_{ji}}{g_{ij}} \right)^{\frac{1}{2}}$ so as to compensate for the biases introduced by the connectivity scheme.

The full MPF objective function in this case is

$$
K = \sum_{j \in \mathcal{D}} \sum_{i \notin \mathcal{D}} g_{ij} \left( \frac{g_{ji}}{g_{ij}} \right)^{\frac{1}{2}} \exp \left[ \frac{1}{2} (E_j - E_i) \right]
$$

(2.67)

where the inner sum is found by averaging over samples from $g_{ij}$. 

44
2.3.9 Continuous State Space Learning with the Connectivity Function Set Via Hamiltonian Monte Carlo

Choosing the connectivity matrix $g_{ij}$ for Minimum Probability Flow Learning is relatively straightforward in systems with binary or discrete state spaces. Nearly any nearest neighbor style scheme seems to work quite well. In continuous state spaces $q \in \mathbb{R}^d$ however, connectivity functions $g(q_i, q_j)$ based on nearest neighbors prove insufficient. For instance, if the non-zero entries in $g(q_i, q_j)$ are drawn from an isotropic Gaussian centered on $q_j$, then several hundred non-zero $g(q_i, q_j)$ are required for every value of $q_j$ in order to achieve effective parameter estimation in some fairly standard problems, such as receptive field estimation in Independent Component Analysis [2].

Qualitatively, we desire to connect every data state $q_j \in D$ to the non data states $q_i$ which will be most informative for learning. The most informative states are those which have high probability under the model distribution $p^{(\infty)}(q)$. We therefore propose to populate $g(q_i, q_j)$ using a Markov transition function for the model distribution. Borrowing techniques from Hamiltonian Monte Carlo [28] we use Hamiltonian dynamics in our transition function, so as to effectively explore the state space.

Extending the state space

In order to implement Hamiltonian dynamics, we first extend the state space to include auxiliary momentum variables.

The initial data and model distributions are $p^{(0)}(q)$ and

$$p^{(\infty)}(q; \theta) = \frac{\exp(-E(q; \theta))}{Z(\theta)}. \quad (2.68)$$

with state space $q \in \mathbb{R}^d$. We introduce auxiliary momentum variables $v \in \mathbb{R}^d$ for each state variable $q$, and call the extended state space including the momentum variables $x = \{q, v\}$. The momentum variables are given an isotropic Gaussian distribution,

$$p(v) = \frac{\exp(-\frac{1}{2}v^Tv)}{\sqrt{2\pi}}, \quad (2.69)$$
and the extended data and model distributions become
\[
p^{(0)}(x) = p^{(0)}(q)p(v) = p^{(0)}(q)\frac{\exp\left(-\frac{1}{2}v^Tv\right)}{\sqrt{2\pi}} \tag{2.70}
\]
\[
p^{(\infty)}(x;\theta) = p^{(\infty)}(q;\theta)p(v) = \frac{\exp\left(-E(q;\theta)\right)\exp\left(-\frac{1}{2}v^Tv\right)}{Z(\theta)} \tag{2.71}
\]
\[
= \frac{\exp\left(-H(x;\theta)\right)}{Z(\theta)\sqrt{2\pi}} \tag{2.72}
\]
\[
H(x;\theta) = E(q;\theta) + \frac{1}{2}v^Tv. \tag{2.73}
\]

The initial (data) distribution over the joint space \(x\) can be realized by drawing a momentum \(v\) from a uniform Gaussian distribution for every observation \(q\) in the dataset \(D\).

### Defining the connectivity function \(g(x_i, x_j)\)

We connect every state \(x_j\) to all states which satisfy one of the following two criteria,

1. All states which share the same position \(q_j\), with a quadratic falloff in \(g(x_i, x_j)\) with the momentum difference \(v_i - v_j\).

2. The state which is reached by simulating Hamiltonian dynamics for a fixed time \(t\) on the system described by \(H(x;\theta_H)\), and then negating the momentum. Note that the parameter vector \(\theta_H\) is used only for the Hamiltonian dynamics.

More formally,
\[
g(x_i, x_j) = \delta(q_i - q_j)\exp\left(-\frac{1}{2}v_i - v_j\right)^2 + \delta(x_i - \text{HAM}(x_j;\theta_H)) \tag{2.74}
\]

where if \(x' = \text{HAM}(x;\theta_H)\), then \(x'\) is the state that results from integrating Hamiltonian dynamics for a time \(t\) and then negating the momentum. Because of the momentum negation, \(x = \text{HAM}(x';\theta_H)\), and \(g(x_i, x_j) = g(x_j, x_i)\).

### Discretizing Hamiltonian dynamics

It is generally impossible to exactly simulate the Hamiltonian dynamics for the system described by \(H(x;\theta_H)\). However, if \(\text{HAM}(x;\theta_H)\) is set to simulate Hamiltonian dynamics via a series of leapfrog steps, it retains the important properties of reversibility
and phase space volume conservation, and can be used in the connectivity function \( g(x_i, x_j) \) in equation [2.76]. In practice, therefore, HAM \((x; \theta_H)\) involves the simulation of Hamiltonian dynamics by a series of leapfrog steps.

**MPF objective function**

The MPF objective function for continuous state spaces and a list of observations \( D \) is

\[
K(\theta; D, \theta_H) = \sum_{x_j \in D} \int g(x_i, x_j) \exp \left( \frac{1}{2} \left[ H(x_j; \theta) - H(x_i; \theta) \right] \right) dx_i.
\]

(2.77)

For the connectivity function \( g(x_i, x_j) \) given previously in this section, this reduces to

\[
K(\theta; D, \theta_H) = \sum_{x_j \in D} \int \exp \left( -||v_i - v_j||_2^2 \right) \exp \left( \frac{1}{2} \left[ \frac{1}{2} v_j^T v_j - \frac{1}{2} v_i^T v_i \right] \right) dv_i + \sum_{x_j \in D} \exp \left( \frac{1}{2} [H(x_j; \theta) - H(HAM(x_j; \theta_H); \theta)] \right).
\]

(2.78)

Note that the first term does not depend on the parameters \( \theta \), and is thus just a constant offset which can be ignored during optimization. Therefore, we can say

\[
K(\theta; D, \theta_H) \sim \sum_{x_j \in D} \exp \left( \frac{1}{2} [H(x_j; \theta) - H(HAM(x_j; \theta_H); \theta)] \right).
\]

(2.79)

Parameter estimation is performed by finding the parameter vector \( \hat{\theta} \) which minimizes the objective function \( K(\theta; D, \theta_H) \),

\[
\hat{\theta} = \arg\min_{\theta} K(\theta; D, \theta_H).
\]

(2.80)
Iteratively improving the objective function

The more similar $\theta_H$ is to $\theta$, the more informative $g(x_i, x_j)$ is for learning. If $\theta_H$ and $\theta$ are dissimilar, then many more data samples will be required in $\mathcal{D}$ to effectively learn. Therefore, we iterate the following procedure, which alternates between finding the $\hat{\theta}$ which minimizes $K(\theta; \mathcal{D}, \theta_H)$, and improving $\theta_H$ by setting it to $\hat{\theta}$,

1. Set $\hat{\theta}^{t+1} = \text{argmin}_\theta K(\theta; \mathcal{D}, \theta_H^t)$
2. Set $\theta_H^{t+1} = \hat{\theta}^{t+1}$

$\hat{\theta}^t$ then represents a steadily improving estimate for the parameter values which best fit the model distribution $p(\infty)(\mathbf{q}; \theta)$ to the data distribution $p(0)(\mathbf{q})$, described by observations $\mathcal{D}$. Practically, step 1 above will frequently be truncated early, perhaps after 10 or 100 L-BFGS gradient descent steps.

2.4 Connection to Other Learning Techniques

2.4.1 Contrastive Divergence

The contrastive divergence update rule can be written in the form

$$\Delta \theta_{CD} \propto -\sum_{j \in \mathcal{D}} \sum_{i \notin \mathcal{D}} \left[ \frac{\partial E_j(\theta)}{\partial \theta} - \frac{\partial E_i(\theta)}{\partial \theta} \right] T_{ij}, \quad (2.81)$$

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 2.81 has obvious similarities to the MPF learning gradient in equation 2.56. Thus, steepest gradient descent under MPF resembles CD updates, but with the MCMC sampling/rejection step $T_{ij}$ replaced by a weighting factor $g_{ij} \exp \left[ \frac{1}{2} (E_j(\theta) - E_i(\theta)) \right]$.

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).
2.4.2 Score Matching

For a continuous state space, MPF reduces to score matching if the connectivity function \( g(x_j, x_i) \) is set to connect all states within a small distance \( r \) of each other,

\[
g(x_i, x_j) = g(x_j, x_i) = \begin{cases} 
0 & d(x_i, x_j) > r \\
1 & d(x_i, x_j) \leq r 
\end{cases},
\]

where \( d(x_i, x_j) \) is the Euclidean distance between states \( x_i \) and \( x_j \). In the limit as \( r \) goes to 0 (within an overall constant and scaling factor),

\[
\lim_{r \to 0} K(\theta) \sim K_{\text{SM}}(\theta) = \sum_{x \in D} \left[ \frac{1}{2} \nabla^2 E(x) \cdot \nabla E(x) - \nabla^2 E(x) \right],
\]

where \( K_{\text{SM}}(\theta) \) is the SM objective function.

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

\[
K_{\text{MPF}} = \frac{1}{N} \sum_{x \in D} \int d^d y \Gamma(y, x) = \frac{1}{N} \sum_{x \in D} \int d^d y \ g(y, x) e^{\frac{1}{2} (E(x|\theta) - E(y|\theta))},
\]

where the sum \( \sum_{x \in D} \) is over all data samples, and \( N \) is the number of samples in the data set \( D \). Now we assume that transitions are only allowed from states \( x \) to states \( y \) that are within a hypercube of side length \( \epsilon \) 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 \( C_\epsilon \), and writing \( y = x + \alpha \) with \( \alpha \in C_\epsilon \), we have

\[
K_{\text{MPF}} = \frac{1}{N} \sum_{x \in D} \int_{C_\epsilon} d^d \alpha \ e^{\frac{1}{2} (E(x|\theta) - E(x+\alpha|\theta))}.
\]
If we Taylor expand in $\alpha$ to second order and ignore cubic and higher terms, we get

$$K_{MPF} \approx \frac{1}{N} \sum_{x \in D} \int_{C_\epsilon} \! d^d \alpha \left( \right)$$

$$- \frac{1}{N} \sum_{x \in D} \int_{C_\epsilon} \! d^d \alpha \left( \frac{1}{2} \sum_{i=1}^d \alpha_i \nabla_x x_i E(x|\theta) \right)$$

$$+ \frac{1}{N} \sum_{x \in D} \int_{C_\epsilon} \! d^d \alpha \left( \frac{1}{4} \left( \sum_{i=1}^d \alpha_i \nabla_x x_i E(x|\theta) \right)^2 \right)$$

$$- \sum_{i,j=1}^d \alpha_i \alpha_j \nabla_x x_i \nabla_x x_j E(x|\theta)) \right).$$

(2.86)

This reduces to

$$K_{MPF} \approx \frac{1}{N} \sum_{x \in D} \left[ \epsilon^d + \frac{1}{4} \frac{1}{2} \epsilon^{d+2} \sum_{i=1}^d \left( \nabla_x x_i E(x|\theta) \right)^2 \right]$$

$$- \frac{1}{12} \epsilon^{d+2} \sum_{i=1}^d \left( \nabla_x^2 x_i E(x|\theta) \right)^2 \right],$$

(2.87)

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

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

(2.88)

$$= K_{SM}.$$

(2.89)

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 \[17\], so the same can be said for MPF in this limit.

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.
2.5 Consistency of the MPF objective function for the exponential family

When an estimator of some statistical parameter $\theta$ converges to the true value of that parameter, the estimator is said to be consistent. More formally, suppose that $p(X|\theta \in \Theta)$ is one of a family of distributions parametrized by $\theta \in \Theta$, and that $X^{\theta} = X_1, X_2, \ldots \ p(X|\theta)$ is an (infinite) sample drawn from $p(X|\theta)$. Let $T_n(X^{\theta})$ be a sequence of estimators, where $T_n$ uses the first $n$ samples from $X^{\theta}$. The sequence $T_n$ is said to be consistent if it converges in probability to the value $\theta$.

In the case of the MPF estimator (i.e. the value of $\theta$ that minimizes $K_{MPF}$), if $K_{MPF}$ can be shown to have a unique global minimum, and if $K_{MPF}$ is minimized at the true value of $\theta$, then it follows that if the gradient of $K_{MPF}$ is zero at a particular value of $\theta$ (the estimated value), this value is the true parameter value.

**Theorem 1.** The MPF objective function is convex for distributions in the exponential family.

*Proof.* The MPF objective function has the form

$$K_{MPF} = \sum_{i \in D} \sum_j g_{ji} \Gamma_{ji} p_i^{(0)}.$$  \hfill (2.90)

It has first derivative with respect to $\theta_n$

$$\frac{\partial K_{MPF}}{\partial \theta_n} = \sum_{i \in D} \sum_j g_{ji} \left( \frac{\partial \Gamma_{ji}}{\partial \theta_n} \right) p_i^{(0)}$$  \hfill (2.91)

$$= \frac{1}{2} \sum_{i \in D} \sum_j g_{ji} \Gamma_{ji} \left( \frac{\partial E_i}{\partial \theta_n} - \frac{\partial E_j}{\partial \theta_n} \right) p_i^{(0)}.  \hfill (2.92)$$

The Hessian is then

$$\frac{\partial^2 K_{MPF}}{\partial \theta_m \partial \theta_n} = \frac{1}{4} \sum_{i \in D} \sum_j g_{ji} \left( \frac{\partial E_i}{\partial \theta_m} - \frac{\partial E_j}{\partial \theta_m} \right) \left( \frac{\partial E_i}{\partial \theta_n} - \frac{\partial E_j}{\partial \theta_n} \right) p_i^{(0)}$$  \hfill (2.93)

$$+ \frac{1}{2} \sum_{i \in D} \sum_j g_{ji} \Gamma_{ji} \left( \frac{\partial^2 E_i}{\partial \theta_m \partial \theta_n} - \frac{\partial^2 E_j}{\partial \theta_m \partial \theta_n} \right) p_i^{(0)}.  \hfill (2.94)$$

The first term in the Hessian is a weighted sum of outer products with positive weights, and is thus positive semi-definite. The second term is zero because of the exponential family assumption: the energy function is linear in the parameters $\theta_n$, and thus has zero second derivatives with respect to them. Since the Hessian of $K_{MPF}$...
is positive semidefinite, $K_{\text{MPF}}$ is convex in the parameters $\theta$.

**Lemma 1.** For distributions in the exponential family, the MPF objective function $K_{\text{MPF}}$ is minimized at the true value of $\theta$ as the sample size goes to infinity.

**Proof.** The gradient of $K_{\text{MPF}}$ with respect to parameters $\theta$ is

$$
\frac{\partial K_{\text{MPF}}}{\partial \theta} = \frac{1}{2} \sum_{i \in D} \sum_{j} g_{ji} \Gamma_{ji} \left( \frac{\partial E_i}{\partial \theta} - \frac{\partial E_j}{\partial \theta} \right) p_i^{(0)}. \tag{2.95}
$$

When the gradient of the MPF objective is evaluated at the true value of the parameter $\theta$ in the limit of infinite data, we can replace $p_i^{(0)}$ with $p_i^{(\infty)}$, since the data will have the exact same distribution as the model. In this case, we can write

$$
\Gamma_{ji} p_i^{(\infty)} = g_{ji} e^{(E_i - E_j)/2} Z(\theta) e^{-E_i} = \frac{g_{ji}}{Z(\theta)} e^{-(E_i + E_j)/2}, \tag{2.96}
$$

and the gradient looks like

$$
\frac{\partial K_{\text{MPF}}}{\partial \theta} = \frac{1}{2} \sum_{i} \sum_{j} \frac{g_{ji}}{Z(\theta)} e^{-(E_i + E_j)/2} \left( \frac{\partial E_i}{\partial \theta} - \frac{\partial E_j}{\partial \theta} \right). \tag{2.97}
$$

This sum over a product of symmetric and antisymmetric quantities is necessarily zero, so the objective function is extremized at the true value of the parameters. In addition, we know that this extremum is a minimum by the convexity assumption justified by Theorem 1.

**Theorem 2.** The MPF estimator is consistent for distributions in the exponential family.

**Proof.** By Theorem 1, any minimum of $K_{\text{MPF}}$ found is the unique global minimum. As the number of points in the sample set increases infinitely, we know by Lemma 1 that this global minimum is at the value of the true parameter vector $\theta$. This concludes the proof.

An graphical illustration of the consistency of the MPF estimator is shown in figure 2.5.

52
Figure 2.5: A plot showing the convergence of parameters to their true values as the number of data points in the sample used for learning increases. This plot was generated by learning an Ising spin glass model from subsets of a large synthetic dataset that increase in sample size. With increasing sample size, the estimator converges to a final value, and the variance of the estimator (indicated by the red region) converges to a minimum (c.f. Cramer-Rao bound).
Chapter 3

Minimum Probability Flow In Practice

In this chapter, we examine the application of minimum probability flow learning to a diverse set of problems, including an Ising spin glass, a deep belief network, and an independent component analysis model (a demonstration of minimum probability flow learning in a continuous state space).

3.1 Ising Model

The Ising model has a long and storied history in physics \cite{7} and machine learning \cite{1} and it has recently been found to be a surprisingly useful model for networks of neurons in the retina \cite{32,33}.

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

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

(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 \cite{38} from a spin glass with known coupling parameters. We used a square 10 \( \times \) 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
Table 3.1: Mean square error in recovered coupling strengths ($\epsilon_J$), mean square error in pairwise correlations ($\epsilon_{corr}$) 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).

<table>
<thead>
<tr>
<th>Technique</th>
<th>$\epsilon_J$</th>
<th>$\epsilon_{corr}$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPF</td>
<td>0.0172</td>
<td>0.0025</td>
<td>$\sim$60</td>
</tr>
<tr>
<td>MFT+TAP</td>
<td>7.7704</td>
<td>0.0983</td>
<td>0.1</td>
</tr>
<tr>
<td>CD-1</td>
<td>0.3196</td>
<td>0.0127</td>
<td>$\sim$20000</td>
</tr>
<tr>
<td>CD-10</td>
<td>0.3341</td>
<td>0.0123</td>
<td>$\sim$20000</td>
</tr>
<tr>
<td>PL</td>
<td>0.0582</td>
<td>0.0036</td>
<td>$\sim$800</td>
</tr>
</tbody>
</table>

Matrix $\Gamma$ had $2^d \times 2^d$ elements, but for learning we populated it sparsely, setting

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

Figure 3.1 shows the mean square error in the estimated $J$ 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 Thouless-Anderson-Palmer (TAP) corrections [39], CD with both one and ten sampling steps per iteration, and pseudolikelihood. 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, and has its global minimum at the true solution (i.e. the Ising model is exponential family, so MPF provides a consistent estimator — see section 2.5). 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.

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) [15] on MNIST handwritten digits. A DBN consists of stacked restricted Boltzmann machines (RBMs), such that the hidden
Figure 3.1: A demonstration of Minimum Probability Flow (MPF) outperforming existing techniques for parameter recovery in an Ising model. (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).
Figure 3.2: 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) Confabulations after training via MPF. A reasonable probabilistic model for handwritten digits has been learned. (c) Confabulations after training via CD. The uneven distribution of digit occurrences suggests that CD-1 has learned a less representative model than MPF.
layer of one RBM forms the visible layer of the next. Each RBM has the form

\[
p^{(\infty)}(x_{\text{vis}}, x_{\text{hid}}; W) = \frac{\exp \left[ x_{\text{hid}}^T W x_{\text{vis}} \right]}{Z(W)},
\]

\[
p^{(\infty)}(x_{\text{vis}}; W) = \frac{\exp \left[ \sum_k \log (1 + \exp \left[ W_k x_{\text{vis}} \right]) \right]}{Z(W)}.
\]

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 \( \Gamma \) was populated so as to connect every state to all states that differed by only a single bit flip (equation 3.2). 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).

Confabulations 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 \( p^{(\infty)}(x_{\text{vis}}|x_{\text{hid}}; W^k) \) for each of the intermediary RBMs, where \( k \) indexes the layer in the stack. 1,000 sampling steps were taken between each confabulation. As shown in Figure 3.2, 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 \( J \in \mathbb{R}^{K \times K} \) of a \( K \) dimensional independent component analysis (ICA) model with a Laplace prior,

\[
p^{(\infty)}(x; J) = e^{-\sum_k |J_k x| / 2K |J|^{-1}},
\]

on 100,000 10 \( \times \) 10 whitened natural image patches from the van Hateren database. Since the log likelihood and its gradient can be calculated analytically for ICA, we solved for \( J \) 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 3.3. MPF minimization was performed by alternating steps of updating the connectivity
Figure 3.3: A continuous state space model fit using minimum probability flow learning (MPF). Learned 10×10 pixel independent component analysis receptive fields $J$ 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.
function $g(x_j, x_i)$ using a Hamiltonian dynamics based scheme, and minimizing the objective function in equation 2.55 via L-BFGS for fixed $g(x_j, x_i)$. This is described in more detail in section 2.3.9.
Chapter 4

Approaching models with unobserved data

It is often advantageous to introduce random variables into the modeling task that don’t correspond to observed samples in a dataset. One example of when this might occur is when the data consist of observations from a sensor network where some sensors fail a percentage of the time. In this case, observed data points will have holes where individual sensors were unavailable. Even though the data is not available, one often still desires a static model with the presence of random variables corresponding to that data. In this case, the variable is said to be latent or hidden.

Another example of a class of models that incorporates unobserved data is hidden Markov models (see figure 4.1). A hidden Markov model consists of a Markov chain of latent states, where at each step, a data point is observed according to a density that is conditional on the latent state.

The Boltzmann machine depicted in Figure 1.2 is yet another example of a model class with latent variables. Since the joint distribution of this class of models is easy to express, this is the one we will use in our examples.

Since the value of a latent variable given a particular data point is expressed

\[
\begin{align*}
&h_{t-1} & h_t & h_{t+1} \\
p(h_t|h_{t-1}) & p(h_t) & p(h_{t+1}|h_t) \\
p(v_t-1|h_{t-1}) & p(v_t|h_t) & p(v_{t+1}|h_{t+1})
\end{align*}
\]

Figure 4.1: A hidden Markov model.
probabilistically, it cannot be treated in the same way as an observed variable. This
makes minimizing the KL divergence between model and data more challenging. One
standard approach in this situation is to use the expectation maximization algorithm.

The expectation maximization (EM) algorithm

The expectation maximization (EM) algorithm alternates between two steps: an
“expectation” step (E-step) and a “maximization” step (M-step). In the E-step,
the expected value of the model’s log likelihood is computed with respect to the
conditional distribution of the latent variables given the observed data. The model
parameters are fixed at their current estimate (which may be a random initialization)
in this step. In the M-step, an updated estimate of the model parameters is computed
by maximizing the expectation value computed in the E-step with respect to the
model parameters. Using $X$ to denote the observed random variables and $Z$ to denote
the latent random variables, we can summarize the EM algorithm as

1. Compute $E_{Z|X}[\ell(D, \theta^t)]$.
2. Compute $\theta^{t+1} = \text{argmin}_\theta E_{Z|X}[\ell(D, \theta)]$.

Step 1 is often accomplished using Gibbs sampling. When these two steps are repeated
until convergence, one is guaranteed to arrive at a local maximum of the log likelihood,
but not necessarily a global maximum. For models in the exponential family the E-
step has a closed form, and the M-step involves maximizing a linear function.

4.1 Approaching latent variable models with MPF

As the objective function for MPF is fundamentally different from that of a maximum
likelihood approach, an EM-like algorithm with all of the same nice properties as the
original is elusive. The most obvious thing to do is to replace the M-step in EM with
a minimization of the MPF objective. The approach becomes:

1. For each observed data vector $x_v$, initialize the hidden units $x_h$ at random.
2. Minimize the MPF objective with the hidden units clamped in their current
   state to obtain a new estimate of the model parameters.
3. Unclamp the hidden units and sample them using the new model parameters.
4. Repeat from step 2 until convergence.

This approach was used to learn a model for the font “3x3”, in which each character
is represented on a binary, three by three grid of pixels (see figure 4.2).
Figure 4.2: The “3x3” font. Letters are represented on a three by three grid of pixels.
Figure 4.3: Samples from a Boltzmann machine model of the letters y, q, g, t, and o.
Figure 4.4: Samples from a Boltzmann machine model of the letters z, q, d, x, and u.
Chapter 5

Conclusion

In this dissertation, 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 (and also potentially with hidden state variables, as demonstrated in this section, although more work in that vein is clearly necessary), including those over both continuous and discrete state space systems. MPF 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 under suitable choices for the dynamics and state space. We hope that this new approach to parameter estimation will enable probabilistic modeling for previously intractable problems.
Bibliography


