Bethe Free Energy and Contrastive Divergence Approximations for Undirected Graphical Models

by

Yee Whye Teh

A thesis submitted in conformity with the requirements
for the degree of Doctorate of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Bethe Free Energy and Contrastive Divergence Approximations for Undirected Graphical Models

Yee Whye Teh
Doctorate of Philosophy
Graduate Department of Computer Science
University of Toronto
2003

As the machine learning community tackles more complex and harder problems, the graphical models needed to solve such problems become larger and more complicated. As a result performing inference and learning exactly for such graphical models become ever more expensive, and approximate inference and learning techniques become ever more prominent.

There are a variety of techniques for approximate inference and learning in the literature. This thesis contributes some new ideas in the products of experts (PoEs) class of models (Hinton, 2002), and the Bethe free energy approximations (Yedidia et al., 2001).

For PoEs, our contribution is in developing new PoE models for continuous-valued domains. We developed RBMrate, a model for discretized continuous-valued data. We applied it to face recognition to demonstrate its abilities. We also developed energy-based models (EBMs) – flexible probabilistic models where the building blocks consist of energy terms computed using a feed-forward network. We show that standard square noiseless independent components analysis (ICA) (Bell and Sejnowski, 1995) can be viewed as a restricted form of EBMs. Extending this relationship with ICA, we describe sparse and over-complete representations of data where the inference process is trivial since it is simply an EBM.

For Bethe free energy approximations, our contribution is a theory relating belief propagation and iterative scaling. We show that both belief propagation and iterative scaling updates can be derived as fixed point equations for constrained minimization of the Bethe free energy. This allows us to develop a new algorithm to directly minimize the Bethe free energy, and to apply the Bethe free energy to learning in addition to inference. We also describe improvements to the efficiency of standard learning algorithms for undirected graphical models (Jiroušek and Přeučil, 1995).

Acknowledgements

I am indebt to many many people throughout these years as a graduate student, for their friendship, patience and kindness, and for making my life both at work and off work as enjoyable as it is. I apologize here first for not mentioning the name of every person I owe thanks to – you know who you are – there are just too many of you.

First and foremost I want to thank Geoff Hinton for being such a wonderful supervisor and for being a great inspiration in my life. Geoff has been very generous, with the time and effort spent in advising me, with financial assistance for my many conference trips, and with the very many humourous jokes he shared with us. In this past few months he has also generously taken time off his busy schedule to read my thesis and give useful comments. I would also like to thank my committee members, Radford Neal, Rich Zemel, Brendan Frey, Fahiem Bacchus and Lawrence Saul for interesting discussion and useful feedback on my thesis.

I have learned tremendously from interacting and collaborating with the wonderful people at both Toronto and Gatsby. In particular I owe many thanks to Max Welling and Quaid Morris for being great collaborators, officemates and friends. Max has been great as an unofficial supervisor and long time collaborator on our Bethe free energy related project, while Quaid has been there through good times and bad. I would also like to thank Quaid Morris and Brian Sallans for proofreading this thesis.

Although doing research is fun, life as a graduate student would have been very dull without the many wonderful friends I have had over the years. I enjoyed very much the wild parties and laid back brunches with the Gatsby bunch while I was in London. I also miss very much the time I spent with Jo, Ben, Gil and other housemates from 57 Exeter Road, especially the monthly pilgrimages to Cafe Bengla. Coming back to Toronto with Geoff, I got to know some great people through MSSA, including Freddy, Darrell, Colin, Jean, Jade, May and Jenn. We did very many fun things together, sometimes I feel like I am slacking off too much. I also enjoyed very much the Saturday hockey and brunches as well as the various festivities at 349 Clinton Street with Quaid, Raja and other friends.

Dedication

For my parents
Teh Ah Pong and Chow Siew Foong
And for my brothers
Yee Neng, Yee Harn and Yee Qin

Contents

1 Introduction **1**

1.1 Outline of Thesis 2

1.1.1 Background 2

1.1.2 Rate-coded Restricted Boltzmann Machines for Face Recognition . . . 3

1.1.3 Energy-based Models for Sparse Overcomplete Representations 3

1.1.4 The Bethe Free Energy for Inference 4

1.1.5 The Bethe Free Energy for Learning 4

2 Background **6**

2.1 Graphical Models 6

2.1.1 Undirected Graphical Models 7

2.1.2 Directed Graphical Models 10

2.1.3 The EM algorithm 11

2.2 Exact Inference 14

2.2.1 Belief Propagation 14

2.2.2 Junction Tree Algorithm 15

2.3 Markov Chain Monte Carlo Sampling 17

2.4 Approximating the Free Energy 20

2.4.1 Variational Approximations 21

2.4.2 Advanced Mean Field Methods 24

2.4.3 Loopy Belief Propagation 29

2.5 Flexible Models with Efficient Exact Inference 31

2.5.1 Maximum Entropy Models 31

2.5.2 Products of Experts 35

2.6 Discussion 39

3 Rate-coded Restricted Boltzmann Machines for Face Recognition 40
3.1 Products of Experts in Continuous-valued Domains . . . 40
3.2 Restricted Boltzmann Machines . . . 41
3.3 Rate-coded Restricted Boltzmann Machines . . . 43
3.4 RBMrate for Facial Modelling . . . 45
3.5 RBMrate for Face Recognition . . . 49
3.5.1 The FERET Database . . . 50
3.5.2 Popular Face Recognition Methods . . . 51
3.5.3 A Face-pair RBMrate Model . . . 52
3.5.4 Comparative Results . . . 55
3.5.5 Receptive Fields Learned by RBMrate . . . 55
3.6 Discussion . . . 57

4 Energy-Based Models for Sparse Overcomplete Representations 59
4.1 Introduction . . . 60
4.2 Square ICA . . . 63
4.2.1 The Causal Generative Approach . . . 64
4.2.2 The Information Maximization Approach . . . 65
4.2.3 Equivalence of the Two Approaches . . . 66
4.2.4 Square ICA with Input Noise . . . 67
4.3 Overcomplete Generalizations of ICA . . . 67
4.3.1 The Causal Generative Approach . . . 67
4.3.2 The Information Maximization Approach . . . 69
4.4 Energy-Based Models . . . 70
4.4.1 Relating EBMs to Causal Generative ICA . . . 72
4.4.2 Relating EBMs to Information Maximization . . . 73
4.5 Parameter Estimation for Energy-Based Models . . . 74
4.5.1 Hybrid Monte Carlo Sampling . . . 78
4.6 Experiment: Blind Source Separation . . . 81
4.7 Experiments: Feature Extraction . . . 84
4.7.1 Speech . . . 85
4.7.2 Natural Image Patches . . . 85
4.7.3 CEDAR Digits . . . 88
4.7.4 FERET Faces . . . 90

4.8 Discussion . . . 94

5 The Bethe Free Energy for Inference **98**

5.1 Introduction . . . 98
5.2 Markov Networks . . . 100
5.3 Ordinary Inference . . . 101
5.3.1 Belief Propagation . . . 102
5.3.2 Bethe Free Energy . . . 103
5.4 Generalized Inference . . . 104
5.4.1 Iterative Scaling . . . 105
5.5 Approximate Generalized Inference . . . 106
5.5.1 Bethe Approximation . . . 107
5.5.2 Relationship to IS and BP . . . 109
5.6 Algorithms to Minimize the Bethe Free Energy . . . 109
5.6.1 Direct fixed point algorithms . . . 110
5.6.2 Constraining the leaves of trees . . . 112
5.6.3 Graphs with cycles . . . 114
5.7 Experiments . . . 116
5.7.1 Speed of Convergence . . . 117
5.7.2 Accuracy of Estimated Marginals . . . 118
5.8 Discussion . . . 119

6 The Bethe Free Energy for Learning **121**

6.1 Introduction . . . 121
6.2 Maximum Entropy . . . 123
6.3 Junction Trees . . . 125
6.4 Unifying Propagation and Scaling . . . 127
6.4.1 Constrained Maximization . . . 127
6.4.2 Efficient Scheduling . . . 129
6.5 Loopy Iterative Scaling . . . 132
6.5.1 Region Graphs . . . 132
6.5.2 Junction Graph Method . . . 133
6.5.3 Cluster Variation Method . . . 135
6.6 Experiment . . . 135
6.7 Discussion . . . 137

7 Discussion **140**

7.1 Products of Experts and Energy-based Models 140

7.2 Bethe Free Energy Approximations 142

Bibliography **144**

List of Algorithms

2.1 EM – Expectation Maximization 12
2.2 Variational EM 24
4.1 Contrastive Divergence Learning for Energy-Based Models 78
4.2 Hybrid Monte Carlo Sampling 80
5.1 Loopy IS – Loopy Iterative Scaling 110
5.2 IS+BP – Iterative Scaling with Loopy Belief Propagation. 111
5.3 UPS-T – Unified Propagation and Scaling on Trees. 115
5.4 UPS – Unified Propagation and Scaling 116
6.1 UPS-JT – Unified Propagation and Scaling for Junction Trees 130

List of Figures

3.1 A restricted Boltzmann machine. 41
3.2 Alternating Gibbs sampling and the terms in the learning rules of a PoE. 43
3.3 Normalizing the face images in the database. 46
3.4 Some examples of processed faces. 46
3.5 The weights learned by RBMrate. Each image is shows the weights adjacent to one hidden unit (called the receptive field). 47
3.6 Reconstructions of faces using the model. In each cell, the left image is the original, while the right one is the reconstruction after one Gibbs sampling iteration. 48
3.7 The weights learned by RBMrate when they are restricted to be non-negative. . 48
3.8 A RBMrate model for pairs of faces. 53
3.9 Error rates of all methods on all test sets. The bars in each group correspond, from left to right, to the rank-1, rank-2, rank-4, rank-8 and rank-16 error rates. The rank-n error rate is the percentage of test images where the n most similar gallery images are all incorrect. 56
3.10 On the left is a test image from $\Delta\mathsf{months}$ and on the right are the 8 most similar images returned by RBMrate. Most human observers cannot find the correct match to the test image. 56
3.11 Example features learned by RBMrate. Each pair of receptive fields constitutes a feature. 57
3.12 Example features learned by RBMratewith non-negative weight constraints. . . 58

4.1 Different methods for non-Gaussian linear components analysis. 61
4.2 Independence properties of three types of models. 62
4.3 (a) Directed graphical model corresponding to the causal generative approach to ICA. (b) Undirected graphical model for an EBM. (c) Directed graphical model representation for an EBM with auxiliary variables clamped at 0. 68

4.4 Mapping used by the information maximization approach given by (4.7). 70

4.5 Evolution of the Amari Distance for the various algorithms on the blind source separation problem, averaged over 100 runs. Note that HMC converged just as fast as the exact sampling algorithm EQUIL, while the exact algorithm EXACT is only slightly faster. The sudden changes in Amari distance are due to the annealing schedule. 83

4.6 Final Amari-Distances for the various algorithms on the blind source separation problem, averaged over 100 runs. The boxes have lines at the lower quartile, median, and upper quartile values. The whiskers show the extent of the rest of the data. Outliers are denoted by "+". 84

4.7 (a) Filters found by the $2\times$ over-complete EBM. The 5 filters in the first row are the ones with largest power, indicating that they represent important features. The 5 filters in the second row are randomly drawn from the remaining 195 filters. (b) Corresponding power-spectra. 86

4.8 Distribution of power over time and frequency for the learned speech features. First the envelope of each filter (the absolute value of its Hilbert transform) was computed and squared. Next, the squared envelope and the power spectrum were thresholded by mapping all values greater than half the peak value to one and the rest to zero. Gaps smaller than 6 samples in time and 3 samples in frequency were filled in. Finally, the outer product of the two "templates" were computed, weighted by the total power of the filter, and added to the diagram. . 87

4.9 Learned filters for natural images. 88

4.10 The spatial layout and size of the filters learned on natural image patches, which are described by the position and size of the bars. 89

4.11 A polar plot of frequency tuning and orientation selectivity of the filters learned on natural image patches, with the centre of each cross at the peak frequency and orientation response, and crosshairs describing the $1/16$-bandwidth. 90

4.12 Learned filters for CEDAR digits. Filters are plotted in whitened space for clarity. 91

4.13 (a) 32 eigenfaces with largest eigenvalue plotted rowwise in descending eigenvalue order. (b) Subset of 32 feature vectors learned using the EBM on ordinary face data. The top row are hand picked, the bottom row randomly selected. (c) Subset of 32 feature vectors learned using architecture I, all randomly selected. 92

4.14 Architecture of a hierarchical non-linear energy-based model. Non-linearities are indicated by sigmoidal units in output layer 1. Energies can be contributed by output variables in both layers, and the number of output variables need not correspond to the number of input variables. 95

5.1 Belief propagation on tree structured Markov networks. (a) $M_{ij}(x_j)$ is the message sent from node i to node j, while $M_{ji}(x_i)$ is the message from j to i. (b) The messages represented by the solid and dotted arrows do not affect one another. 102

5.2 Scheduling propagation updates for UPS-T. The observed nodes are grey while the hidden nodes are white. The only propagation updates required between scaling updates at nodes 1 and 2 are those on the path from node 1 to node 2 (the arrows). 113

5.3 Scheduling scaling updates for UPS-T. Updates are performed in a depth first manner on the tree. The numbers describe the order in which nodes are visited. Black arrows are forward traversals and white arrows are backtracking traversals. 114

5.4 Clamping hidden nodes of a graph to make it singly connected. Hidden nodes are white; observed nodes are grey; and clamped nodes are black. The little nodes are the replicas. 116

5.5 Speed of convergence of the various algorithms. The box lines are at the median and upper and lower quartiles, and the whiskers describe the extent of data. An algorithm or subroutine is considered converged if the beliefs change by less than 10^{-10}. 118

5.6 Each plot shows the mean absolute errors for various settings of σ_w (x-axis) and σ_b (y-axis). The top plots show errors for loopy IS and bottom plots show errors for UPS. The inset shows the cases (black) when loopy IS did not converge within 2000 iterations, with linear damping slowly increasing to 0.99. When loopy BP did not converge the errors are calculated from the current beliefs after 2000 iterations. 119

6.1 An ordering satisfying the running intersection property to distribute the iterative scaling change at c_1 to the rest of the graph. 126

6.2 (a) Shafer-Shenoy propagation updates. (b) Computing clique distributions from messages. (c) Computing separator distributions from messages. 128

6.3 The dashed lines are the messages which are still correct, while the solid line denotes the message that is updated to become correct. 131
6.4 (a) An example of a graphical model. (b) A junction graph for the model. Notice that the separator on the bottom only contains 8 and not 5. (c) A region graph constructed using the cluster variational method. 134
6.5 Some examples of the images of '8's. The images have been binarized by thresholding. 136
6.6 (a) Average log likelihoods over training data as a function of junction *graph* clique size. Each curve is averaged over all models with a certain maximal clique size of the corresponding junction *tree*. For models of treewidth 9, we also plotted the standard deviation of the log likelihoods. The circles denote the log likelihood under the exact maximum likelihood parameters (i.e. when the junction graph coincides with the junction tree). (b) Average L_1 distances of the various models. The notations are the same as in (a). 137

Chapter 1

Introduction

Machine learning researchers are faced with the difficult task of creating algorithms that learn to interpret, understand, and otherwise make good use of data collected from the world, just as people do. There are three main types of learning tasks, resulting in three different classes of algorithms.

In *supervised* learning, models are trained to extract specific information from their inputs by making use of teaching signals which associate a desired output with each input. The model is trained to predict the desired outputs from the inputs. Some currently popular supervised learning methods are multi-layer perceptrons, support vector machines and Gaussian processes.

In *reinforcement* learning, we have agents which can act upon the world. Rewards are given to agents depending on how desirable the current state of the world is, and agents learn to act in such a way as to maximize their long-term rewards. Reinforcement learning can be viewed as a semi-supervised learning paradigm, since agents are told how well are they doing, but not actually what to do.

In both supervised and reinforcement learning, a teaching signal is provided to the system. The teaching signal defines a goal for the system: to agree with the teacher or to maximize expected reward. In *unsupervised* learning, no such signal is provided, and models simply have to "make sense" of the world based on their observations. This generally means to extract certain invariances or infer some hidden causes which could have given rise to the observations.

In recent years a principled approach to unsupervised learning has been generally adopted in which a probabilistic model of the world is constructed such that actual observations get high probability. This thesis follows this modelling approach to unsupervised learning and uses graphical models.

In addition to machine learning, graphical models have been studied in a variety of other fields, including statistics and applied probability, data compression and communication, and graph theory. To cope with the ever more complex problems that graphical models have been applied to, many approximate methods for learning and inference in graphical models have been proposed. In this thesis, we describe the contributions to approximate learning and inference that we have made.

There are two parts to this thesis: products of experts (PoEs), and Bethe free energy approximations. For PoEs, our contribution is in developing new models for continuous-valued domains. This led to applications of PoEs to face recognition, and to an interesting and novel extension of independent components analysis to the over-complete case. For Bethe free energy approximations, our contribution is in developing a new theory connecting belief propagation and iterative scaling. This will allow us to develop algorithms for minimizing the Bethe free energy and to learn graphical models using Bethe free energy approximations.

1.1 Outline of Thesis

1.1.1 Background

In chapter 2 we describe the many popular techniques for approximate inference and learning. This will serve as a backdrop in which we can place the contributions of this thesis. We first describe directed and undirected graphical models and some related theory, e.g. exponential families and the EM algorithm in section 2.1. Then we described exact inference algorithms, in particular belief propagation and the junction tree algorithm in section 2.2. In section 2.3 we describe Markov chain Monte Carlo sampling for inference. Section 2.4 describes the three major classes of approximate inference algorithms: variational approximations (section 2.4.1), ad-

vanced mean field methods (section 2.4.2) and loopy belief propagation (section 2.4.3). Since all three can be understood as approximating the free energy, there are many interesting links between them. When dealing with partially observed directed graphical models, an approximate inference algorithm translates directly into an approximate learning algorithm using the EM algorithm. Section 2.5 deals instead with undirected graphical models. When these are fully observed inference is trivial but learning is still hard due to the partition function. We describe maximum entropy models in section 2.5.1 and products of experts in section 2.5.2.

1.1.2 Rate-coded Restricted Boltzmann Machines for Face Recognition

In chapter 3 we describe a new product of experts model for continuous domains. In particular we describe an extension of restricted Boltzmann machines so that they can handle discretized continuous values. In section 3.2 we describe restricted Boltzmann machines in detail and the extension to discretized continuous values. Then we apply these rated-coded restricted Boltzmann machines to face modelling in section 3.4 and to face recognition in section 3.5. We show that rate-coded restricted Boltzmann machines are comparable to other popular models for face recognition. However, unlike these other models, there is plenty of room for further development to improve recognition accuracy.

1.1.3 Energy-based Models for Sparse Overcomplete Representations

Rate-coded restricted Boltzmann machines are a good step towards modelling continuous values with products of experts, but they are limited since they can only model discretized values. In chapter 4 we describe a new class of products of experts called energy-based models which will naturally handle continuous values. Energy-based models have an interesting relationship with independent component analysis and we will take independent component analysis as a starting point in chapter 4. In section 4.1 we give an overview of this relationship including the three views of independent component analysis. We show how all three views reduce to the same model in the square noiseless case in section 4.2, and how they differ in

the over-complete case in sections 4.3 and 4.4. We discuss learning in energy-based models using contrastive divergence in section 4.5 and show in section 4.6 that contrastive divergence learning gives good results as compared to exact methods. Section 4.7 gives some simulation results showing energy-based models extracting interesting features.

1.1.4 The Bethe Free Energy for Inference

In particular, chapter 5 deals with unified propagation and scaling, an algorithm for directly minimizing the Bethe free energy. In section 5.4 we describe generalization of inference based on minimizing a KL divergence subject to certain observational constraints. Then in section 5.5 we describe approximations for generalized inference based on the Bethe free energy and show an interesting relationship between belief propagation and iterative scaling. In section 5.6 we describe various algorithms to minimize this Bethe free energy approximation for generalized inference, culminating in unified propagation and scaling. Finally in section 5.7 we describe some experiments comparing unified propagation and scaling to loopy belief propagation.

1.1.5 The Bethe Free Energy for Learning

In chapter 5 we described a theory relating belief propagation and iterative scaling, and used it to derive an algorithm to minimize the Bethe free energy for inference. In chapter 6 we make use of the fact that iterative scaling is a standard algorithm to learn undirected graphical models and show how we can apply the same theory to learn undirected graphical models. In section 6.2 we describe the maximum entropy framework and its relationship to maximum likelihood learning of undirected graphical models. We also derive iterative scaling as an exact learning algorithm for undirected graphical models. Then in section 6.3 we describe improvements to the efficiency of iterative scaling based on junction trees. Using the ideas developed in chapter 5 we propose in section 6.4 unified propagation and scaling for junction trees, an algorithm which further improves the efficiency of iterative scaling on junction trees. When this is still not enough to make the learning tractable, in section 6.5 we propose approximations based on

region-graph free energies and using loopy iterative scaling to optimize these approximate free energies. In section 6.7 we show loopy iterative scaling working on a simple problem.

Chapter 2

Background

Graphical models have been one of the most significant advances in machine learning in the last decade. Given a graphical model, there are two operations one typically performs: inference, and learning. Because both are typically costly operations, a variety of schemes have been proposed in the literature to either approximate or sidestep them. We survey some of these here.

2.1 Graphical Models

In recent years graphical models have gained prominence as a viable class of probabilistic models suitable for unsupervised learning (Jordan, 1998). Graphical models are graphs with probabilistic semantics. Nodes of the graph represent variables, and the structure of the graph describes conditional independencies between the variables. In particular, the lack of an edge between two nodes means they are conditionally independent given observations at some subset of nodes. There can be visible nodes representing observations, and hidden nodes representing unobserved causes in the world. Further, the graphs could have directed or undirected edges, giving rise to the two classes of graphical models we shall describe later in the section.

Given a graphical model, there are two main operations one performs on it. One can train the graphical model to assign high probability to observations, and one can infer the posterior

distribution over hidden variables given some observations. For learning, we assume that the graphical structure is fixed, and only fine tune the parameters to assign high probability to observations. One can also learn the structure of the graphical model, or use a Bayesian approach, where we average over structures and/or parameters (Cooper and Herskovits, 1992, Lam and Bacchus, 1994, Heckerman, 1998, Friedman, 1998, Beal and Ghahramani, 2003). For inference, we typically do not need the posterior distribution explicitly, but rather the expectation of certain functions under the posterior distribution (and perhaps some non-linear combinations of these expectations). For example, if the functions are delta functions indicating the value of a subset of variables, the expectations become marginals of the posterior.

Learning and inference in graphical models are intimately tied. The EM algorithm, to be introduced in section 2.1.3, and its variants are the most common methods of learning graphical models when only a subset of the variables are observed. Part of the algorithm involves inferring the posterior distribution over hidden nodes given observations. For undirected graphical models, looser notions of learning and inference can be seen as dual to each other (see Wainwright, 2002). We will explore the relationship between them further in the contexts of directed and undirected graphical models.

In the following two subsections we will describe both undirected and directed graphical models. To keep the exposition simple we will deal with the completely observed case (that is, every random variable is observed during learning) first. The partially observed case is described in the next subsection on the EM algorithm.

2.1.1 Undirected Graphical Models

In the following we will take x to be the set of variables and $\widetilde{P}(x)$ to be an empirical distribution from which we wish to learn the parameters of the undirected graphical model. $\widetilde{P}(x)$ is typically an average over a number of delta functions, each given by a particular training sample.

Undirected graphical models represent affinities between nodes with undirected edges. Let

c be a clique[1] of the graph, and let the potential $\psi_c(x_c)$ be a non-negative function of the states x_c in the clique. The probability distribution is given as

$$P(x) = \frac{1}{Z} \prod_c \psi_c(x_c) \tag{2.1}$$

where the product is over all cliques c of the undirected graph and Z is a normalization constant called the partition function.

There is a set of conditional independency statements which the undirected graphical model expresses. Let A, B and C be three disjoint sets of variables. If every path from a node in A to a node in B contains a node in C, we say C separates A from B. The conditional independencies expressed by the undirected graphical model are then exactly

$$\left\{ A \perp\!\!\!\perp B \;\middle|\; C \text{ for all disjoint subsets } A, B \text{ and } C \text{ such that } C \text{ separates } A \text{ and } B \right\} \tag{2.2}$$

The Hammersley-Clifford theorem relates the family of distributions given by (2.1) (as we vary the potential functions ϕ_c) to the set of conditional independencies expressed by the undirected graphical model (2.2) (Clifford, 1990). In particular, every distribution expressed by (2.1) satisfies (2.2). Conversely, if a distribution $P(x)$ satisfies (2.2) *and* satisfies $P(x) > 0$ for every x, then $P(x)$ can be expressible as (2.1) for some settings of the potentials. The converse is slightly weaker as there are certain degenerate distributions which satisfy (2.2) but not (2.1) (see Lauritzen, 1996).

Suppose that each potential function $\psi_c(x_c)$ has parameters θ_c. Using (2.1), the expected log likelihood is given by

$$E_{\tilde{P}}[\log P(x)] = \sum_c E_{\tilde{P}}[\log \psi_c(x_c)] - \log Z \tag{2.3}$$

To learn the parameters, we maximize the expected log likelihood with respect to θ_c. Differentiating (2.3) with respect to θ_c, we get

$$\frac{\partial E_{\tilde{P}}[\log P(x)]}{\partial \theta_c} = E_{\tilde{P}} \left[\frac{\partial \log \psi_c(x_c)}{\partial \theta_c} \right] - E_P \left[\frac{\partial \log \psi_s(x_c)}{\partial \theta_c} \right] \tag{2.4}$$

[1] A clique is a totally connected subset of nodes.

where the second term is obtained by differentiating $\log Z$:

$$\log Z = \log \sum_{x'} \prod_{c} \psi_c(x'_c) \tag{2.5}$$

The functions $\frac{\partial \log \psi_c(x_c)}{\partial \theta_c}$ are usually easy to compute. If the cliques c are small enough, this means that the first term is easy as well – just average over the training data (note that we are dealing with the completely observed case here). However the second term is much trickier as it requires an expectation over $P(x)$, which is often intractable. In such a case, the normal routine is to take samples from $P(x)$ and approximate the second term by averaging over the samples (Hinton and Sejnowski, 1986, Zhu et al., 1997). In chapter 6 we describe another approach based on loopy belief propagation. Note that calculating the second term can be construed as inference, since it is an expectation over the model distribution $P(x)$, which can be taken as the "posterior" when there is no observation. This is the first instance of learning requiring inference we shall encounter. The next is for the EM algorithm in section 2.1.3

In many cases of interest, the log potential functions $\psi_c(x_c)$ are linear in the parameters θ_c[2]:

$$\log \psi_c(x_c) = \theta_c^T s_c(x_c) \tag{2.6}$$

where $s_c(x_c)$ is a fixed vector-valued function. Then we have

$$P(x) = \frac{1}{Z} \exp\left(\sum_c \theta_c^T s_c(x_c)\right) \tag{2.7}$$

so the undirected graphical model describes an exponential family with sufficient statistics functions s_c and natural parameters θ_c's (although the s_c's may not be linearly independent so this representation is not minimal). The learning rule (2.4) simplifies to

$$\frac{\partial E_{\tilde{P}}[\log P(x)]}{\partial \theta_c} = E_{\tilde{P}}[s_c(x_c)] - E_P[s_c(x_c)] \tag{2.8}$$

Setting the derivatives to zero, we see that learning attempts to match up the sufficient statistics of the model $E_P[s_c(x_c)]$ to the sufficient statistics of the data $E_{\tilde{P}}[s_c(x_c)]$, which, in this case, can always be attained because the log likelihood is concave in the parameters θ_c's.

[2]Note however that the products of experts to be described in section 2.5.2 do not have this nice property.

Note that we can understand learning as the process of estimating the natural parameters θ_c's given the sufficient statistics $E_{\tilde{P}}[s_c(x_c)]$'s, while inference is essentially computing the sufficient statistics $E_P[s_c(x_c)]$'s from a given set of natural parameters θ_c's. Hence learning and inference are "dual" to each other in the case of exponential families. This duality is a simple consequence of a beautiful theory of exponential families called information geometry. We will not elaborate here but refer the reader to Amari and Nagaoka (2000) and Gupta (1987), and Wainwright (2002, chap. 2) for a brief but excellent introduction. Another aspect of this duality, between maximum likelihood and maximum entropy, is explored further in section 2.5.1 and in chapter 6.

2.1.2 Directed Graphical Models

Directed graphical models, or causal models, represent cause and effect relationships between nodes using directed edges – an edge $u \to v$ means that the parent u is a (direct) cause of v. The overall probability distribution over all the nodes x is

$$P(x) = \prod_i P(x_i|pa_i) \tag{2.9}$$

where pa_i are the parents of node x_i.

The set of conditional independencies represented by directed graphical models is slightly more complex than for undirected ones. Let A, B and C again be three disjoint sets of nodes. We say C *d-separates* A and B if for every node v on every path from a node in A to a node in B, one of the following holds:

- if both edges containing v on the path point inwards to v, then neither v nor any of its descendants are in C;
- otherwise, v is in C.

The set of conditional independencies represented by the directed graphical model is:

$$\left\{ A \perp\!\!\!\perp B \;\middle|\; C \text{ for all disjoint subsets } A, B \text{ and } C \text{ such that } C \text{ d-separates } A \text{ and } B \right\} \tag{2.10}$$

Again we can show that the set of distributions satisfying the conditional independencies (2.10) is exactly the distributions having the form (2.9) (without the need for being strictly positive this time). Notice that individual distributions in this family could satisfy other conditional independencies in addition to those given in (2.10).

Suppose that each conditional distribution $P(x_i|pa_i)$ is parameterized by θ_i. The derivative of the log likelihood with respect to θ_i is

$$\frac{\partial E_{\widetilde{P}}[\log P(x)]}{\partial \theta_i} = E_{\widetilde{P}}\left[\frac{\partial \log P(x_i|pa_i)}{\partial \theta_i}\right] \tag{2.11}$$

Because there is no partition function involved, learning in directed graphical models is much simpler. In addition, each derivative in (2.11) depends on a single θ_i, so solving for the θ_i's are decoupled. Note however that this is only true in the completely observed case. In the next subsection we describe the partially observed case, where dependencies between the parameters are introduced by the unobserved variables and an iterative scheme (the EM algorithm) is needed to optimize them.

2.1.3 The EM algorithm

In the last two sections we discussed learning in graphical models when all the variables are observed. However we often build models where certain variables are always unobserved or hidden, both for modelling reasons — we wish to discover the hidden causes of the observed data — and because this increases the modelling flexibility. A simple example illustrating both reasons is mixture models. Here we will discuss learning in graphical models (both directed and undirected) from partially observed data through the EM algorithm (Dempster et al., 1977, Neal and Hinton, 1998).

We assume that the nodes are partitioned into two sets x and y with y being the visible or observed variables, and x the hidden or unobserved variables. We have an empirical distribution $\widetilde{P}(y)$ given by the training set, and a model distribution $P(x, y|\theta)$ over $\{x, y\}$ with parameters θ.

The log probability of generating the observations is

$$\mathcal{L}(\theta) = E_{\widetilde{P}}[\log P(y|\theta)] = \int \log P(y|\theta)\, \widetilde{P}(y)\, dy \tag{2.12}$$

The EM algorithm, given in algorithm 2.1, is an iterative procedure which at each iteration starts with an initial θ and produces a better estimate $\theta^{(\text{new})}$.

Algorithm 2.1 EM – Expectation Maximization

1. Initialize θ to some values.
2. Repeat until convergence criterion has been met:
3. **Expectation (E) step**: This is the inference step.

   Fills in the unobserved variables x using the current posterior distribution $Q(x|y) = P(x|y, \theta)$. This is normally done independently for each training observation y.
4. **Maximization (M) step**: This is the learning step.

   Set $\theta^{(\text{new})}$ to maximize the complete data likelihood, assuming that the current posterior is correct

$$\theta^{(\text{new})} = \underset{\theta'}{\operatorname{argmax}} \int \log P(x, y|\theta')\, Q(x|y) \widetilde{P}(y)\, dx\, dy \tag{2.13}$$

Instead of maximizing the complete data likelihood at each M step, one can find a $\theta^{(\text{new})}$ which only improves upon the likelihood. This is called the generalized EM (GEM) algorithm. The EM and GEM algorithms were shown to never decrease the log likelihood, $\mathcal{L}(\theta^{(\text{new})}) \geq \mathcal{L}(\theta)$. Assuming that $\mathcal{L}$ is bounded above and other technical conditions like differentiability apply, it can be further shown that the algorithm will converge to a local maximum of $\mathcal{L}$ (Dempster et al., 1977).

For simple models like linear Gaussian models and tree-structured graphical models, the E step can be exactly and efficiently carried out. For more complex models, we need to approximate the inference in the E step. One possibility is to approximate it using samples from

the posterior that are obtained using Markov chains (see section 2.3). Deterministic approximations for inference are given in section 2.4, while in section 2.5 we describe two complex models which allow exact inference. For directed graphical models, the M step is usually simple, while it is much harder for undirected graphical models due to the partition function. As a result more sophisticated algorithms like iterative proportional fitting (IPF) and generalized iterative scaling (GIS) discussed in section 2.5.1 are required. The contrastive divergence approach in section 2.5.2 sidesteps this problem by approximately optimizing a different function than the log likelihood.

The theory behind the EM algorithm was generalized by Neal and Hinton (1998), where EM is viewed as coordinate minimization of the variational free energy[3]

$$\mathcal{F}_{var}(Q,\theta) = \int \Big[-\log P(x,y|\theta) + \log Q(x|y) \Big] Q(x|y)\widetilde{P}(y)\, dx\, dy \tag{2.14}$$

The E step minimizes $\mathcal{F}_{var}(Q,\theta)$ with respect to Q and the M step minimizes $\mathcal{F}_{var}(Q,\theta)$ with respect to θ. The E step is normally performed independently for each training observation y expressed by $\widetilde{P}$.

$\mathcal{F}_{var}(Q,\theta)$ is always an upper bound on $-\mathcal{L}(\theta)$, with equality exactly at $Q(x|y) = P(x|y,\theta)$ almost everywhere on the support of $\widetilde{P}(y)$. This can be easily seen by rearranging:

$$\mathcal{F}_{var}(Q,\theta) = \int \left[-\log P(y|\theta) + KL\big(Q(x|y) \| P(x|y,\theta)\big) \right] \widetilde{P}(y)\, dy \tag{2.15}$$

with the last term being the KL divergence from $Q(x|y)$ to $P(x|y,\theta)$:

$$KL\left(q(x)\|p(x)\right) = \int q(x) \log \frac{q(x)}{p(x)}\, dx \tag{2.16}$$

In this formulation, analogous to generalized M steps, we can now use generalized E steps which update the approximate posterior $Q(x|y)$ to lower $KL\big(Q(x|y)\|P(x|y,\theta)\big)$.

The wake-sleep algorithm used to learn the Helmholtz machine (Dayan et al., 1995) and the recognition networks of Morris (Morris, 2002) use a similar idea. Both parameterize $Q(x|y)$

[3]We follow the physics convention of free energies being minimized rather than the one in Neal and Hinton (1998) where it is negated and maximized.

using a single set of parameters for all y and improve the approximation at each step using gradient descent on the reverse KL divergence.

When the minimization over all possible distributions $Q(x|y)$ is intractable, we can restrict $Q(x|y)$ to come from a tractable family of distributions, resulting in a further lower bound. A larger family will give a smaller KL divergence hence a better bound, as seen from (2.15). This is the starting point for variational approximations which will be discussed in section 2.4.1.

$\mathcal{F}_{var}(Q, \theta)$ is analogous to the free energy from statistical physics. This link has enabled many methods developed in the statistical physics community to be used in neural computations, and, recently with generalized belief propagation, vice versa. Some of these methods will be discussed in sections 2.4.2 and 2.4.3.

2.2 Exact Inference

In this section we describe algorithms for doing exact inference on graphical models. We start with the simple case for when the graphical model is a tree. This is the belief propagation algorithm in section 2.2.1. Then we generalize this to the junction tree algorithm for general graphical models in section 2.2.2.

2.2.1 Belief Propagation

Belief propagation (BP) is an exact inference algorithm for tree-structured graphical models (Pearl, 1988). The trees can be either directed or undirected, but for the rest of this section we shall deal only with undirected trees. Directed trees can be easily converted into undirected ones by simply dropping the directionality of the edges[4]

Suppose we have a tree G. By i, j and k we shall denote vertices of G and by (ij) we shall mean an edge of G. For each i let x_i be the random variable associated with i and $\hat{x}_i$ be a state of x_i. Let $\psi_i(x_i)$ and $\psi_{ij}(x_i, x_j)$ be the marginal and pairwise potentials of the tree-structured

[4]For general directed graphs, including polytrees (i.e. directed trees where there is more than one root), we need to moralize the graph before dropping the directionality, i.e. add edges between any two parents of a node.

undirected graphical model so that

$$P(x) \propto \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \psi_i(x_i) \tag{2.17}$$

For each edge (ij) let $M_{ij}(\hat{x}_j)$ be the message from i to j and vice versa for $M_{ji}(\hat{x}_i)$. BP iteratively updates the messages with the rules:

$$M_{ij}^{(\text{new})}(\hat{x}_j) = \sum_{\hat{x}_i} \psi_{ij}(\hat{x}_i, \hat{x}_j)\psi_i(\hat{x}_i) \prod_{k \in N(i)\backslash j} M_{ki}(\hat{x}_i) \tag{2.18}$$

where $N(i)\backslash j$ indicates all neighbours of i except j. If the message updates are ordered efficiently, each only needs to be updated once for convergence. If n is the number of edges, this is only $2n$ updates. For example, the forward-backward algorithm for HMMs is just BP applied to a chain, with the messages updated in a forward and a backward pass, and it converges after both passes. The beliefs can be obtained from the messages as

$$b_i(x_i) \propto \psi_i(\hat{x}_i) \prod_{j \in N(i)} M_{ji}(\hat{x}_i) \tag{2.19}$$

$$b_{ij}(x_i, x_j) \propto \psi_{ij}(x_i, x_j)\psi_i(x_i)\psi_j(x_j) \prod_{k \in N(i)\backslash j} M_{ki}(x_i) \prod_{l \in N(j)\backslash i} M_{lj}(x_j) \tag{2.20}$$

At convergence, we can show that the beliefs are exactly the marginal distributions of $P(x)$.

BP can also be applied to graphical models with cycles. In particular, the algorithm we have just described can be applied directly to pairwise Markov networks. A pairwise Markov network is an undirected graphical model where we take only the edges as cliques, so that the form of the distribution is exactly (2.17). However loopy BP for pairwise Markov networks is an approximate algorithm. In section 2.4.3 we elaborate on what exactly this approximation is.

2.2.2 Junction Tree Algorithm

For general graphical models the junction tree algorithm is a popular algorithm for exact inference (Jensen, 1996, Cowell, 1998). In this section we shall briefly describe the construction of junction trees as well as the propagation algorithms which are run on the junction trees. We will not prove any of the very nice graph theoretic results relating to junction trees. Please see Jensen (1996) and Cowell (1998) for more information.

Directed graphical models must first be converted into undirected ones by moralizing. Essentially, this means adding edges among the parents of each node, and dropping the directionality of the original edges. The form of the distribution of the directed graphical model (2.9) stays the same as for the undirected one (2.1). The moralization simply makes sure that each term in (2.9) corresponds to a clique in the undirected model.

Now given an undirected graphical model, we collect nodes into clusters such that the topology of the clusters forms a junction tree. That is, each edge of the graph has to be in at least one cluster, and there is a tree with nodes labelled by the clusters such that for any two clusters their intersection is a subset of every cluster on the path (on the tree) between them. This is typically obtained by eliminating nodes one at a time. For each node in an elimination ordering, we add edges among the uneliminated neighbours of the node so that they form a clique. After eliminating all the nodes, the maximal cliques of the resulting graph forms the clusters of the junction tree.

Given a junction tree we can now collect the potentials of the original graphical model into potentials $\phi_c(x_c)$ for each cluster of the junction tree such that the distribution (2.1) is equivalent to

$$P(x) = \frac{1}{Z} \prod_c \phi_c(x_c) \tag{2.21}$$

where now the product is over the clusters of the junction tree.

There are a number of propagation algorithms for junction trees. Foremost among them are Shafer-Shenoy (SS) (Shafer and Shenoy, 1990) and Hugin (Jensen, 1996). SS propagation is a direct generalization of BP to junction trees. For any two neighbouring clusters c_1 and c_2 in the junction tree with separator s, there are messages $M_{sc_2}(x_s)$ and $M_{sc_1}(x_s)$ going from c_1 to c_2 and back. The messages are updated with the SS propagation rules

$$M_{sc_2}(x_s) \leftarrow\propto \sum_{x_{c_1 \setminus s}} \phi_{c_1}(x_{c_1}) \prod_{s' \neq s} M_{s'c_1}(x_{s'}) \tag{2.22}$$

where the product is over all separators neighbouring c_1 except s itself. We can again schedule the updates so that each message needs only be updated once. At convergence, the marginal

distributions are exactly the beliefs

$$b_c(x_c) \propto \phi_c(x_c) \prod_s M_{sc}(x_s) \qquad\qquad b_s(x_s) \propto M_{sc_1}(x_s) M_{sc_2}(x_s) \tag{2.23}$$

Hugin propagation, on the other hand, updates the beliefs directly. We initalize $b_c(x_c) = \phi_c(x_c)$ and $b_s(x_s) = 1$. Let c_1 and c_2 again be neighbouring clusters and s the separator. The updates are:

$$b_{c_2}(x_{c_2}) \leftarrow\propto b_{c_2}(x_{c_2}) \frac{b_{c_1}(x_s)}{b_s(x_s)} \qquad\qquad b_s(x_s) \leftarrow\propto b_{c_1}(x_s) \tag{2.24}$$

Again there are schedules which require only twice the number of clusters of updates for convergence. In fact, we can show that SS propagation and Hugin propagation are equivalent, by identifying the beliefs and messages using (2.23).

The computational complexity of the junction tree algorithm is exponential in the size of the largest cluster. If this is much smaller than the graph itself the computational speed-up is significant. Unfortunately, for many graphical models of interest the cluster sizes are still too large to be practical. Loopy BP and its generalizations are natural extensions for efficient but approximate inference (Yedidia et al., 2001, 2002).

Jiroušek and Přeučil (1995) proposed an algorithm for learning undirected graphical models using essentially the junction tree algorithm in an inner loop. In chapter 6 we describe a generalization of their algorithm using loopy BP ideas.

2.3 Markov Chain Monte Carlo Sampling

We can do approximate inference in graphical models using samples from the posterior distributions. These distributions are often complex and multi-dimensional, and they typically have most of their probability mass concentrated in very small regions of the space. As a result simple Monte Carlo sampling schemes are not suitable. Instead Markov chain Monte Carlo (MCMC) sampling is often the only viable sampling method for these distributions. MCMC sampling is very versatile and applicable in virtually all circumstances as there is a wide range of methods to choose from. It is also asymptotically correct if we are prepared to wait long

enough for the Markov chains to converge and to obtain a large number of samples. However it is often computationally very expensive leading to various issues and trade-offs. There is a very large literature on MCMC sampling, and I will only introduce a few issues and methods that are relevant to neural computation. Refer to Neal (1993) and Gilks et al. (1996) for thorough reviews.

In MCMC sampling, an ergodic Markov chain with transition conditional distribution $T(x'|x)$ is constructed such that the equilibrium distribution is the desired posterior distribution $P(x|y,\theta)$. Starting from a simple initial distribution $x_0 \sim Q_0(x_0)$, the Markov chain is simulated by sampling from $x_t \sim T(x_t|x_{t-1})$ during each time step t. Let

$$Q_t(x') = \int T(x'|x)\, Q_{t-1}(x)\, dx \tag{2.25}$$

be the distribution of x_t. Assume that $Q_t(x)$ for every t and $P(x|y,\theta)$ are absolutely continuous with respect to some measure μ. Since T is ergodic, $Q_t(x) \rightarrow P(x|y,\theta)$ almost everywhere with respect to μ as $t \rightarrow \infty$.

Independent samples are desirable to reduce the variance of the sampler. To get independent samples, one can run multiple Markov chains to convergence, and get one sample from each of them. But this is wasteful because the Markov chains often take so long to converge. Instead more than one sample is often taken from each Markov chain. In fact it is better to use all samples from each Markov chain after the initial burn-in period. The number of chains to run, the length of the burn-in period, and the number of samples to obtain from each chain all have to be optimized empirically.

For multi-dimensional distributions, because most of the state space has negligible probability, the Markov chain can only make small random changes to the state at each step. If the distribution is multi-modal, it is often very hard for the Markov chain to move from one mode to another, since the Markov chain will have to move through regions of low probability separating the modes. There are a few methods of tackling this problem, including simulated annealing (Kirkpatrick et al., 1983, Neal, 2001), simulated tempering (Marinari and Parisi, 1992, Neal, 1996), Metropolis-coupled MCMC sampling (Geyer, 1991), and entropic sampling (Lee, 1993). Markov chains often use small step sizes to move around the state space.

As a result they often exhibit random walk behaviour. During a random walk, the distance travelled grows only as the square root of the time spent. Hence even moving within a mode requires a significant amount of time. Hybrid Monte Carlo methods and over-relaxation (Neal, 1993) have been proposed to reduce random walk behaviour.

As a result of multiple modes and random walk behaviour, the time required for the Markov chain to converge to the equilibrium distribution (the mixing time) is often quite long. Also we often do not know the mixing time of the chain, and any upper bound we can derive is typically impracticably large. Many methods of detecting convergence have been proposed, but none of them is perfect. Surprisingly, recently developed techniques based on coupling can generate exact samples from the equilibrium distribution of Markov chains (Propp and Wilson, 1996). Getting exact samples eliminates the problem of convergence detection. However they are only efficient for Markov chains satisfying a certain monotonicity property. Bounding chains have to be used for non-monotonic chains, including most Markov chains used in machine learning. As a result exact sampling methods have not found widespread use in machine learning. However, methods based on coupling have recently made their way into useful techniques in MCMC sampling (Neal, 2002).

The sequence of distributions Q_t defined in (2.25) can be interpreted as approximations to the posterior distribution which converges to the posterior as $t \to \infty$. In fact, it can be shown that $\mathcal{F}_{var}(Q_{t+1}, \theta) \leq \mathcal{F}_{var}(Q_t, \theta)$ for each t (Sallans, 1998). Hence each MCMC step can be thought of as generalized E steps. Of course we do not really compute the Q_t's, we can only obtain samples from Q_t. If gradient ascent is used for the M step, then a useful way of thinking about sampling is as part of a stochastic gradient ascent M step, where the gradient of the parameters is estimated using samples from the approximate posterior Q_t, while the generalized E step updates the approximate posterior from Q_t to Q_{t+1} exactly. Taken in this light, samples after every MCMC step can be used in estimating the parameters. Further, the parameters can be updated even when the chain has not converged. This is a much more efficient use of computational resources. The overall algorithm is guaranteed to improve $\mathcal{F}_{var}$ stochastically, and so long as the learning rate is small and decreases slowly to 0, the parameters

will converge to ML solutions. This is called brief Gibbs sampling (Hinton et al., 1998).

A drawback of the above algorithm is that MCMC samples have to be stored for every training case, and it cannot easily be generalized to online learning. Hinton *et al* proposed instead an even briefer Gibbs sampling, where at each E step we start the Markov chain from a fixed state, simulate the Markov chain for a few steps, and take samples, ignoring whether the chain has converged. Since the E steps do not have to improve $\mathcal{F}_{var}$, and the M step does not take into account changes to the approximate posterior when parameters are changed, this algorithm is not guaranteed to stochastically improve $\mathcal{F}_{var}$. However in practice it tends to do better than the original brief Gibbs sampling, as the variance of the samples is smaller[5]. The contrastive divergence learning algorithm for products of experts in section 2.5.2 can be viewed as another improvement on this scheme.

MCMC techniques are useful when we need to evaluate expectations with respect to the posterior distribution in an (asymptotically) unbiased manner. Because they are asymptotically unbiased and applicable in almost all circumstances, they are often the preferred choice for statisticians. However, due to the computational resources required to get enough samples to reduce the variance to an acceptable level, after running the chain for enough steps to reduce the bias to an acceptable level, they are often not seen as practical for inference and learning in machine learning. Techniques like brief Gibbs sampling, however, are efficient enough to make MCMC practical for learning.

2.4 Approximating the Free Energy

There is a variety of approximation methods which can be understood as first approximating some (intractable) free energy, then minimizing the resulting approximate free energy. This includes variational approximations, the advanced mean field methods from statistical physics, and loopy belief propagation. We discuss each in detail in the following subsections.

[5]Personal communication from B. Sallans.

2.4.1 Variational Approximations

Recall the variational free energy from section 2.1.3

$$\mathcal{F}_{var}(Q,\theta) = \int \Big[-\log P(x,y|\theta) + \log Q(x|y) \Big] Q(x|y)\widetilde{P}(y)\, dx\, dy \tag{2.26}$$

$$= \int \Big[-\log P(y|\theta) + KL\Big(Q(x|y) \| P(x|y,\theta)\Big) \Big] \widetilde{P}(y)\, dy \tag{2.27}$$

$\mathcal{F}_{var}(Q,\theta)$ is an upper bound on the negative log likelihood, and EM is coordinate minimization of $\mathcal{F}_{var}(Q,\theta)$ with respect to Q and θ.

When $\mathcal{F}_{var}$ is intractable to work with, we can upper bound it further by introducing additional auxiliary variables so that the resulting expression is tractable (Jordan et al., 1998). This is called variational approximation. The algorithms derived using variational approximations are deterministic and often orders of magnitude faster than those using MCMC sampling. Since they improve an upper bound on $\mathcal{F}_{var}$, convergence is guaranteed and can be easily detected. They are however not guaranteed to improve the likelihood at each step – the likelihood could improve or the bound could get tighter.

Often, the posterior distribution $P(x|y,\theta)$ is intractable, so it is expensive to minimize $\mathcal{F}_{var}(Q,\theta)$ with respect to Q. However, we can minimize $\mathcal{F}_{var}(Q,\theta)$ with respect to Q assuming that Q comes from a tractable family of distributions. This is called the block variational approximation[6]. From (2.27), we see that a tighter bound is equivalent to having a family of distributions which approximates the posterior well in terms of the KL divergence between them. In the rest of this section we shall deal with the block variational approach.

If $P(x,y|\theta)$ is a graphical model, then the posterior distribution can be defined by the undirected graphical model G obtained by moralizing the original graphical model, and conditioning out the observed nodes. The moralized graph is often dense; hence computing $P(x|y,\theta)$ is intractable. We can approximate the posterior by taking Q to be a graphical model obtained by removing edges from G until it becomes tractable[7]. In this case the assumptions imposed on Q

[6] Jordan et al. (1998) calls this the block variational method, even though it is the more commonly used one, to distinguish it from the approach as described in the previous paragraph.

[7] Although sometimes the best distribution may not be obtained by removing edges.

are exactly the additional conditional independencies expressed by the removal of the edges of the graphical model. Further, the less edges we remove, the tighter the resulting bound will be.

The simplest approximation is obtained by removing all edges from G, i.e. assuming that all the hidden variables are independent of each other given the observations:

$$Q(x) = \prod_i Q_i(x_i) \tag{2.28}$$

This is called the naive mean field (MF) approximation, and is the most popular variational approximation as it is the simplest to work with. However, since it removes all edges from the posterior, it is the worst approximation we can get by removing edges. Also, it cannot handle any explaining away effects as these are essentially dependencies among the hidden variables conditioned on the observations. The MF approximation is further elaborated from a statistical physics point of view in section 2.4.2.

One obtains a better bound by retaining some edges of the posterior graphical model so that the resulting distribution is still tractable. Further, by being careful about which edges to retain, the most coupled hidden variables can still be dependent in the approximation, and the bound on $\mathcal{F}_{var}$ will be tighter. This is termed structured variational approximation.

Looking at (2.27), we note that minimizing $\mathcal{F}_{var}$ with respect to Q is equivalent to decreasing the KL divergence from Q to $P(x|y,\theta)$. Since the KL divergence is not symmetric, this is not equivalent to decreasing $KL(P(x|y,\theta)\|Q(x|y))$. For MF approximations, this means that the optimal $Q(x_i|y)$ will not be the marginal posterior $P(x_i|y,\theta)$. In general, the reversed KL divergence means that it is more costly to have $P(x|y,\theta)$ small when $Q(x|y)$ is large than the reverse. If the posterior consists of many well separated modes but Q is a single-modal distribution, then Q will try to model a single mode of the posterior well rather than capture all of them. In certain applications it is more important to capture all the modes. For example, in medical diagnosis applications, where observations are symptoms, and hidden variables are diseases, it is better to be conservative and report all possible disease occurrences. In these cases, there are more suitable approximations which try to decrease the other KL divergence (Frey et al., 2001, Minka, 2001, Kappen and Rodriguez, 2001, Morris, 2002).

From (2.27) again, we see that minimizing $\mathcal{F}_{var}(Q, \theta)$ with respect to θ can be interpreted as maximizing $\mathcal{L}(\theta)$ with a regularizing term

$$\int KL(Q(x|y)\|P(x|y,\theta))\, \widetilde{P}(y)\, dy \tag{2.29}$$

which encourages $P(x|y,\theta)$ to be as close to $Q(x|y)$ as possible. Since $Q(x|y)$ is often a simpler distribution, $P(x|y,\theta)$ will tend to be simpler in the same way. On the one hand, this is good as it encourages the model to have simple posteriors and will be a control on the complexity of the learned model. On the other hand the resulting distribution will not be as accurate as it will try to ignore certain dependencies between hidden variables.

An exciting recent advance has been the development of variational Bayesian learning (Ghahramani and Beal, 2000). Given a prior distribution $P(\theta)$ over θ and observations $\mathbf{y} = \{y^{(c)}\}$, correct Bayesian learning is to compute the posterior over θ

$$P(\theta|\mathbf{y}) = \frac{P(\mathbf{y}|\theta)P(\theta)}{\int P(\mathbf{y}|\theta)P(\theta)\, d\theta} \tag{2.30}$$

After learning, the probability of some observation y' is given by

$$P(y'|\mathbf{y}) = \int P(y'|\theta)P(\theta|\mathbf{y})\, d\theta \tag{2.31}$$

Because the posterior over $\mathbf{x} = \{x^{(c)}\}$ and over θ are coupled and the parameters of a model are often unidentifiable (e.g. the components of a mixture model can be permuted without affecting the probabilities), exact Bayesian learning is often intractable even for simple models like mixtures of Gaussians. One approach is to approximate the whole posterior $P(\theta|\mathbf{y})$ with a single maximum a posteriori (MAP) estimate of θ. This is what the EM algorithm gives. However if the distribution $P(x, y|\theta)$ is in an exponential family with θ being the natural parameters, and the prior over θ is conjugate to $P(x, y|\theta)$[8], one can use a variational approximation in which θ and $\mathbf{x}$ are independent. The variational free energy is then

$$\mathcal{F}_{var}(Q(\mathbf{x}), Q(\theta)) = \int \Big[-\log P(\mathbf{x}, \mathbf{y}, \theta) + \log Q(\mathbf{x})Q(\theta) \Big]\, Q(\mathbf{x})Q(\theta)\, d\mathbf{x}\, d\theta \tag{2.32}$$

and the variational EM algorithm is given in algorithm 2.2.

[8]The prior $P(\theta)$ is conjugate to $P(x, y|\theta)$ if it is of the same functional form as $P(x, y|\theta)$ in terms of θ.

Algorithm 2.2 Variational EM

1. Initialize $Q(\theta)$ to some simple distribution.
2. Repeat until convergence criterion has been met:
3. **Variational E step**: Maximize $\mathcal{F}_{var}$ with respect to $Q(\mathbf{x})$:

$$Q(\mathbf{x}) \propto \exp\left(E_{Q(\theta)}\left[\log P(\mathbf{x}, \mathbf{y}, \theta)\right]\right) \tag{2.33}$$

4. **Variational M step**: Maximize $\mathcal{F}_{var}$ with respect to $Q(\theta)$:

$$Q(\theta) \propto \exp\left(E_{Q(\mathbf{x})}\left[\log P(\mathbf{x}, \mathbf{y}, \theta)\right]\right) \tag{2.34}$$

The advantage of this method is that the computational cost of variational EM is often on the same order as normal EM, and in practice it is only marginally more expensive than the normal EM algorithm. The variational E step can often be done by performing a normal E step using a single pseudo-observation of θ; while the variational M step can be done by simply extracting certain sufficient statistics. When propagation algorithms like the Kalman filter are needed for inference, they can be generalized to the variational case as well (Ghahramani and Beal, 2001). With such modest additional costs, the advantages are enormous. One has a lower bound on the evidence for a model. This can be used to compare models and perform model selection. The resulting model is simply more flexible and powerful than if only a single estimate of θ is used. We can also avoid over-fitting if we integrate out parameters such that the number of hyperparameters is independent of the model size.

2.4.2 Advanced Mean Field Methods

In section 2.4.1 we introduced the naive MF approximation as a variational approximation where the posterior is assumed to be factorized. The term “mean field” originated in statistical physics where it roughly means to replace the fluctuating “field” surrounding a unit by its mean value. The field surrounding a unit is the influence the other units have on the unit. The naive MF approximation is the simplest of a wide spectrum of methods in statistical physics. In this

section we shall describe a few of the methods developed which have been applied to inference in graphical models. For further information please refer to Saad and Opper (2001).

To illustrate the ideas while keeping notation simple, we shall focus on developing MF methods for inference in Boltzmann machines (BM). Boltzmann machines are undirected graphical models where the potentials only involve one or two nodes. Let i and j denote nodes of the graph, $x_i \in \{0, 1\}$ be the variable associated with i, $W_{ij} \in \mathbb{R}$ be a weight connecting nodes i and j, and $b_i \in \mathbb{R}$ be the bias of node i. The probability of a state $x = \{x_i\}$ is defined as

$$P(x) = \frac{1}{Z} \exp\left(\sum_{(ij)} W_{ij} x_i x_j + \sum_i b_i x_i\right) \tag{2.35}$$

where Z is the partition function.

Suppose a number of nodes O were observed, with $x_k = \hat{x}_k$ for each $k \in O$. Recall that the variational free energy is

$$\mathcal{F}_{var}(Q, \beta) = E_Q\left[-\sum_{(ij)} \beta W_{ij} x_i x_j - \sum_i \beta b_i x_i + \log Q(x)\right] \tag{2.36}$$

For notational simplicity, we have defined $Q(x_k) = \delta(x_k - \hat{x}_k)$ for $k \in O$ and omitted references to θ and the constant $\log Z$. We have instead added in the inverse temperature β for use later. When it is not mentioned, we shall assume that $\beta = 1$.

For the naive MF approximation, we assume that Q factorizes, and suppose that the mean of node i is $E_Q[x_i] = m_i$. Set $m_k = E_Q[x_k] = \hat{x}_k$ for each $k \in O$ and let $m = \{m_i\}$. Then the naive MF free energy is

$$\mathcal{F}_{mf}(m) = -\sum_{(ij)} W_{ij} m_i m_j - \sum_i b_i m_i + \sum_i \left(m_i \log m_i + (1-m_i)\log(1-m_i)\right) \tag{2.37}$$

Minimizing $\mathcal{F}_{mf}(m)$ with respect to m_i, we get the MF fixed point equations

$$m_i^{(\text{new})} = \text{sigmoid}\left(\sum_{j \neq i} W_{ij} m_j + b_i\right) \tag{2.38}$$

Suppose we assume that the mean values are $E_Q[x_i] = m_i$ for each node i, but we do not assume anything else about Q, then the minimum attainable $\mathcal{F}_{var}$ is called the Gibbs free

energy,

$$\mathcal{F}_{gibbs}(m,\beta) = \min\{\mathcal{F}_{var}(Q,\beta) \,:\, E_Q[x_i] = m_i \,\forall i\} \tag{2.39}$$

We wish to minimize $\mathcal{F}_{gibbs}(m,1)$ with respect to m. Since the value of m that minimizes $\mathcal{F}_{gibbs}$ is the exact posterior marginals $m_i = P(x_i|y)$, minimizing $\mathcal{F}_{gibbs}(m,1)$ is not usually tractable. Instead, the Gibbs free energy is important because it is fertile territory for approximation schemes.

In particular, we can expand $\mathcal{F}_{gibbs}(m,\beta)$ in terms of β around $\beta = 0$, and evaluate the expansion at 1. This is called Plefka's expansion (Plefka, 1982).

$$\mathcal{F}_{gibbs}(m,1) = \mathcal{F}_{gibbs}(m,0) + \frac{\partial}{\partial\beta}\mathcal{F}_{gibbs}(m,0) + \frac{1}{2!}\left(\frac{\partial}{\partial\beta}\right)^2 \mathcal{F}_{gibbs}(m,0) + \cdots \tag{2.40}$$

Deriving the terms of the expansion requires quite an amount of ingenuity, but the first three terms can readily be obtained

$$\mathcal{F}_{gibbs}(m,0) = \sum_i \left(m_i \log m_i + (1-m_i)\log(1-m_i)\right) \tag{2.41}$$

$$\frac{\partial}{\partial\beta}\mathcal{F}_{gibbs}(m,0) = -\sum_{(ij)} W_{ij} m_i m_j - \sum_i b_i m_i \tag{2.42}$$

$$\frac{1}{2}\left(\frac{\partial}{\partial\beta}\right)^2 \mathcal{F}_{gibbs}(m,0) = -\frac{1}{2}\sum_{(ij)} W_{ij}^2 m_i(1-m_i)m_j(1-m_j) \tag{2.43}$$

We see that the first two terms correspond to $\mathcal{F}_{mf}$, and we can also understand how the naive MF approximation is bad – it is simply a linear approximation to the true free energy. Including the third term, one obtains the TAP free energy $\mathcal{F}_{tap}$ (Thouless et al., 1977). Setting the derivatives of $\mathcal{F}_{tap}$ with respect to m_i to 0, we get the TAP equations

$$m_i = \text{sigmoid}\left(\sum_{j\neq i} W_{ij} m_j + b_i + \frac{2m_i - 1}{2}\sum_{j\neq i} W_{ij}^2 m_j(1-m_j)\right) \tag{2.44}$$

The extra term at the right is called the Onsager reaction term. $\mathcal{F}_{tap}$ is no longer a lower bound on $\mathcal{F}_{var}$ and the TAP equations (2.44) are not guaranteed to converge. However, if they do converge they tend to converge to better estimates of the true posterior marginals than the MF equations (2.38). It is conceivable that extra terms can be derived and used to obtain better estimates of the posterior marginals. We shall view the Bethe approximation in section 2.4.3 in

this light. For directed graphical models, when the Gibbs free energy is not in a particularly nice form to work with, similar approaches have been used to expand other quantities of interest like KL divergences, and variants of the TAP equations have been derived (Tanaka, 2001, Kappen and Rodriguez, 2001).

Plefka's expansion can be used to understand the contribution of different substructures of the BM toward the Gibbs free energy and posterior distribution. For each term appearing in Plefka's expansion, a diagram can be drawn to summarize the contribution of the term as follows: a vertex is drawn for each index i appearing in the term, and an edge is drawn connecting vertices i and j for each W_{ij} that appears in the term. It can be easily seen that the terms appearing in Plefka's expansion are those whose diagrams are subgraphs of the BM (treating multiple edges connecting two vertices as a single edge). Less obviously, these terms must also be strongly irreducible, i.e. the corresponding diagram cannot be split into two on removing a vertex (Georges and Yedidia, 1991, Welling and Teh, 2003). Hence Plefka's expansion for a tree can only contain relatively simple terms of the form $f(m)W_{ij}^n$ for some n and function $f(m)$; and cycles make inference hard in BMs because they introduce complex terms into the Gibbs free energy. We shall revisit this matter in section 2.4.3 when we try to understand how loopy belief propagation works.

A different MF method is called the linear response correction. The initial observation is that we can study the system represented by our model by studying its perturbations as we vary the parameters of the system. In particular, consider varying the biases b_i about its original value, say b_i^0:

$$b_i = b_i^0 + \theta_i \tag{2.45}$$

where θ_i are small variations. Express both the distribution $P_\theta(x)$ from (2.35) and its partition function Z_θ as functions of the variations $\theta = \{\theta_i\}$. Then by direct differentiation, we get

$$\frac{\partial \log Z_\theta}{\partial \theta_j} = E_{P_\theta}[x_j] \tag{2.46}$$

$$\frac{\partial^2 \log Z_\theta}{\partial \theta_i \partial \theta_j} = \frac{\partial E_{P_\theta}[x_j]}{\partial \theta_i} = E_{P_\theta}[x_i x_j] - E_{P_\theta}[x_i]E_{P_\theta}[x_j] \tag{2.47}$$

Setting $\theta = 0$ above shows that we recover the first two cumulants of $P_0(x) = P(x)$ from

the derivatives of $\log Z_\theta$. In other words, the log partition function is the cumulant generating function of $P(x)$, our desired distribution[9]. Using any MF approximation, we obtain estimates m_i of $E_P[x_i]$, and we can use (2.47) to approximate the correlation between two hidden nodes:

$$\frac{\partial m_i}{\partial b_j} \approx \frac{\partial E_P[x_i]}{\partial b_j} = E_P[x_i x_j] - E_P[x_i]E_P[x_j] \tag{2.48}$$

For example, using the naive MF approximation for a BM, we differentiate (2.38) to get

$$\Big[E_P[x_i x_j] - E_P[x_i]E_P[x_j]\Big] \approx \left(\Big[\frac{\delta_{ij}}{m_i(1-m_i)} - W_{ij}\Big]\right)^{-1} \tag{2.49}$$

where $[f(i,j)]$ is a matrix with $(ij)^{th}$ element given by $f(i,j)$. The linear response correction has been used in fully visible BM learning where the weights and biases can be estimated directly from the empirical pairwise correlations and marginals using a single matrix inversion (Kappen and Rodríguez, 1998).

Other MF methods that has been applied to machine learning are the field theoretic method, the cavity method, and the Bethe approximation. For an introduction to these methods refer to (Opper and Winther, 2001). Loopy belief propagation and its relation to the Bethe approximation will be discussed in section 2.4.3.

In summary, it is important to remember that many of these methods are not guaranteed to converge, and except for the simplest MF approximation, there is also no bound on the variational free energy that we can compute. However, if one were to be careful about using them only when they converge, they will often give estimates of the posterior that are more accurate than the simplest MF approximation. For example, the TAP and Bethe approximations to be introduced in section 2.4.3 have been shown to frequently converge when the weights are small and there is high evidence, i.e. many nodes are observed. Most of these advanced MF methods are not well-suited as part of a learning algorithm like EM, but are more suitable when more accurate estimates of the posterior marginals are required, and when convergence is more likely.

[9]Technically the cumulant generating function is the logarithm of the moment generating function, $K(\theta) = \log E_P\left[e^{\theta^T x}\right]$. It is easy to see that $K(\theta) = \log Z_\theta - \log Z_0$ with $\log Z_0$ constant.

2.4.3 Loopy Belief Propagation

In this section we shall be discussing loopy belief propagation (loopy BP). Loopy BP is just the algorithm given by the BP iterations (2.18) applied without alteration to graphs with cycles. Unlike the junction tree algorithm, loopy BP is an approximate inference algorithm that is not guaranteed to converge. However when it does converge it often converges quickly and the approximation is typically quite accurate (Murphy et al., 1999). Loopy BP has been applied successfully to error-correcting codes (turbo-decoding) (McEliece et al., 1998), and machine vision (Freeman and Pasztor, 1999). As a matter of fact turbo-decoding was discovered first, and the realization that it is loopy BP on an appropriate graph spurred research into the behaviour of loopy BP itself.

Initial experimental results showed that loop BP, when it converges, often produce estimates of the posterior marginal distributions that are more accurate than those obtained with naive MF methods (Weiss, 1997, Murphy et al., 1999). These experiments also showed that loopy BP often converges for graphs with longer, fewer, and less strongly coupled cycles. A theory based on equal amounts of "double counting" was developed which shows that loop BP should give the correct MAP estimates for a certain class of "balanced networks" (Weiss, 2000). This was further elaborated by linking the unrolled networks to Gibbs measures on computation trees (Tatikonda and Jordan, 2002).

However, a breakthrough was made in understanding loopy BP when Yedidia et al. (2001) discovered that fixed points of loopy BP are exactly the stationary points of the Bethe free energy. Using the notation of section 2.2.1, the Bethe free energy is

$$\begin{aligned} \mathcal{F}_{bethe} = &- \sum_{(ij)} \sum_{x_i,x_j} b_{ij}(x_i,x_j) \log \psi_{ij}(x_i,x_j) - \sum_i \sum_{x_i} b_i(x_i) \log \psi_i(x_i) \\ &+ \sum_{(ij)} \sum_{x_i,x_j} b_{ij}(x_i,x_j) \log b_{ij}(x_i,x_j) + \sum_i (1-n_i) \sum_{x_i} b_i(x_i) \log b_i(x_i) \end{aligned} \tag{2.50}$$

where n_i is the number of neighbours of node i. The beliefs b_i and b_{ij} need to satisfy the marginalization constraints $\sum_{x_j} b_{ij}(x_i,x_j) = b_i(x_i)$ and $\sum_{x_i} b_i(x_i) = 1$. Note that these constraints do not imply that the beliefs are marginal distributions of a single overall distribution.

They can be expressed by adding Lagrange multipliers to (2.50). The messages of loopy BP are related to the Lagrange multipliers, and the message updates (2.18) is just a set of consistency equations that has to be satisfied at the stationary points of $\mathcal{F}_{bethe}$. These consistency equations are derived by setting the derivatives of the Lagrangian[10] with respect to the beliefs to zero. Experimentally, the stable fixed points of loopy BP are the local minima of $\mathcal{F}_{bethe}$. Heskes (2003) has recently proven that they are in fact local minima.

$\mathcal{F}_{bethe}$ is an approximation to the Gibbs free energy $\mathcal{F}_{gibbs}$ given in (2.39). Consider $\mathcal{F}_{bethe}$ and $\mathcal{F}_{gibbs}$ for a BM. Since $\mathcal{F}_{bethe}$ is exact for a tree, we can deduce that for general BMs $\mathcal{F}_{bethe}$ consists of exactly all terms in Plefka's expansion of the form $f(m)W_{ij}^n$. This makes exact the relationship between loopy BP and other advanced MF methods like TAP. It also explains the finding that loopy BP performs better on graphs with fewer and longer cycles with weakly coupled nodes. On these graphs, there are less terms in Plefka's expansion which are not in $\mathcal{F}_{bethe}$ and these terms tend to be smaller (since they involve a product $\prod_{(ij)} W_{ij}$ with each term being small), hence $\mathcal{F}_{bethe}$ is a good approximation to $\mathcal{F}_{gibbs}$.

Since we want to minimize the Gibbs free energy, and the Bethe free energy $\mathcal{F}_{bethe}$ is an approximation to the Gibbs free energy, it makes sense to try to minimize the Bethe free energy itself. Loopy BP is just one of a variety of methods to minimize it. Belief Optimization (BO) is one such method which directly minimizes $\mathcal{F}_{bethe}$ with respect to the beliefs at every iteration (Welling and Teh, 2001). BO works only for Boltzmann machines. In chapter 5 we describe UPS, another algorithm guaranteed to minimize $\mathcal{F}_{bethe}$. Unlike BO, UPS works for general pairwise Markov networks. The stable nature of BO and UPS makes them more suitable as part of a learning algorithm.

Welling and Teh (2001) also showed that in the Boltzmann machine case, $\mathcal{F}_{bethe}$ and $\mathcal{F}_{tap}$ are equivalent up to second order, but $\mathcal{F}_{bethe}$ has more higher order terms. This suggests that the Bethe approximation is more accurate than the TAP approximation. This was confirmed

[10]There are two standard usages of the term "Lagrangian", one in classical mechanics to describe the movements of masses and one in constrained optimization where it is a constructed objective which includes both the original objective to be minimized as well as the constraints. Here as well as through out this thesis we shall use the second meaning, which is the more common one in the machine learning community.

experimentally when they showed that BO always performs better than TAP.

When there are strongly coupled clusters of nodes with loops, the Bethe approximation will be bad. We can account for the errors introduced by the clusters by using beliefs spanning the clusters (i.e. $b_c(x_c)$ where c is a cluster) and adding extra terms to the approximation. This generalization of the Bethe free energy is called the Kikuchi free energy. The larger the clusters used, the more accurate is the Kikuchi approximation. If we are willing to use very large clusters, the Kikuchi free energy can be made exact. However, in many models, we know that the strongly coupled clusters are just locally connected units (e.g. neighbouring pixels in images) so in practice only small clusters are needed for a good approximation. We can derive a generalized BP algorithm whose fixed points are the stationary points of the Kikuchi free energy (Yedidia et al., 2001, Yedidia, 2001). Even using reasonably small clusters, generalized BP has been shown to converge more often and gives more accurate estimates of the beliefs (Yedidia et al., 2001).

2.5 Flexible Models with Efficient Exact Inference

Rather than starting with a model which has intractable inference and learning, another possibility is to start with a model with efficient exact inference, and sidestep the need for approximate inference techniques. These are the maximum entropy models and the products of experts, to be discussed in the next two subsections. The disadvantage of such models is that we have to deal with the partition function during learning. For maximum entropy models, the approaches have been to either restrict its application to models where we can calculate the partition function by brute force, or to use MCMC sampling. For products of experts, the idea is to use an approximate learning method called contrastive divergence.

2.5.1 Maximum Entropy Models

Maximum entropy is a principle for probabilistic modelling which states that if one knows nothing about a process, then one should introduce the least distortion in modelling the process

by using the simplest distribution. This can be achieved by using a distribution with maximum entropy, which, on a compact state space, is the uniform distribution. If one does know something about the modelled process in terms of certain constraints on the distribution, then the distribution to use should be one which is as uniform as possible, i.e. with maximum entropy, subject to the constraints being satisfied. We shall deal in particular with constraints expressed as expectations of some functions under the process. Not all constraints can be represented this way, but the maximum entropy distributions under such constraints are exactly the exponential families. Many commonly encountered models are in fact exponential families.

Let $\widetilde{P}(x)$ be the empirical distribution which we wish to model. Let a vector-valued function $s(x)$ be a vector of features of x, and let $S = E_{\widetilde{P}}[s(x)]$. The constraint on the distribution $P(x)$ is expressed as $E_P[s(x)] = S$. Using Lagrange multipliers, it is easy to show that the maximum entropy distribution $P(x)$ has the form

$$P(x) = \frac{1}{Z} \exp\left(\theta^T s(x)\right) \tag{2.51}$$

where the Lagrange multipliers θ are chosen so that $E_P[s(x)] = S$, and Z is the partition function. Note that to determine $P(x)$ we only require knowledge of S. As we vary S, the maximum entropy distributions form exactly the exponential family with sufficient statistics functions $s(x)$, sufficient statistics S and natural parameters θ. Further, the θ which satisfy the constraints are in fact the maximum likelihood parameters for the exponential family (2.51) for the empirical distribution $\widetilde{P}$. This duality between maximum entropy and maximum likelihood is an important aspect of information geometry that we referred to in section 2.1.1.

Popular exponential families like Gaussians and Dirichlets use very simple features $s(x)$ for constraints. To model more complex processes, one can extend the distribution to have hidden variables, for example, using graphical models like sigmoid belief networks. Note however that if the hidden variables are integrated out the model will in general not be in the exponential family. Hidden variables are useful for other purposes, for example to discover and model hidden causes of observations, but inferring posterior distributions over them is often intractable. If we only want to model the process well in terms of high likelihood, another possibility is simply to use more complex features, and more of them. This is the approach

taken in maximum entropy modelling (MaxEnt).

Note that $s(x)$ can be any arbitrarily complex deterministic function of x, and inference will still be trivial – just evaluate $s(x)$[11]. Each feature of $s(x)$ can even be the log likelihood of x under a probability distribution with hidden variables. So long as each distribution allows efficient inference, inferring the overall distribution over all hidden variables is still efficient. This is the products of experts approach in the next section.

To apply maximum entropy for probabilistic modelling, there are two problems to solve. The first is to find a good set of features to use (Della Pietra et al., 1997, Zhu et al., 1997). The log probability of the training data is

$$E_{\tilde{P}}[\log P(x)] = E_{\tilde{P}}\left[\theta^T s(x)\right] - \log Z = E_P\left[\theta^T s(x)\right] - \log Z = E_P\left[\log P(x)\right] \tag{2.52}$$

which is just the negative entropy of $P(x)$. Hence in the limit of large training set, where over-fitting is not a concern, we would like to choose a feature set such that the maximum entropy distribution has minimum entropy (maximum likelihood). Zhu et al. (1997) called this the minimax entropy principle. This is usually achieved using a double loop algorithm where one finds the maximum entropy θ in the inner loop, and greedily adds a feature which maximizes the likelihood in the outer loop. To avoid over-fitting, a preference for simpler features is usually used in the greedy search.

The second problem in maximum entropy modelling is to determine the parameters θ which satisfy the constraints. The good news here is that the log likelihood $E_{\tilde{P}}[\log P(x)]$ is a convex function of θ and there is a unique global maximum where $E_P[s(x)] = S$. The bad news is that due to the partition function, it can be expensive to integrate over $P(x)$ and to determine θ. The simplest method to optimize θ is by gradient ascent. From (2.51),

$$\frac{\partial E_{\tilde{P}}[\log P(x)]}{\partial \theta} = E_{\tilde{P}}[s(x)] - E_P[s(x)] = S - E_P[s(x)] \tag{2.53}$$

S can easily be computed from data. The second term, coming from the partition function, requires an expectation over $P(x)$. This is normally intractable hence an approximate inference

[11] We typically can only evaluate the log likelihood up to a constant due to the partition function Z, but this is usually enough for most purposes. Also, by inference here we do not mean calculating expectations under $P(x)$.

procedure will be required to estimate the expectations. As $P(x)$ is normally highly multimodal, techniques like variational approximation are not suitable (another problem with typical variational techniques is that they upper bound rather than lower bound the log likelihood in this case). Gibbs sampling along with simulated annealing is often the only applicable method, as the conditional distribution of each variable in x given the others can often be computed easily. In chapter 6 we introduce an approximate method to learn MaxEnt models by approximating the free energy.

However, gradient ascent is troublesome as the learning rate has to be tweaked for performance, and different learning rates might be required for different features and regions of θ. Improved iterative scaling (IIS) is a general procedure to determine θ without the need to set parameters like the learning rate (Della Pietra et al., 1997). However, IIS still needs to sample from $P(x)$ to estimate $E_P[s(x)]$ (or needs to calculate it exactly instead). We assume for the following that $s(x) \geq 0$ for all x.

At each iteration of IIS, each θ_i is updated by an amount δ_i where δ_i is the unique solution to the following equation:

$$E_P\left[s_i(x) \exp\left(t(x)\delta_i\right)\right] = S_i \tag{2.54}$$

where $t(x) = \sum_i s_i(x)$ is the sum of components of $s(x)$. The equations (2.54) for the δ_i's are decoupled and so each δ_i can be solved independently from the rest. If $t(x) = T$ is a constant over x, (2.54) can be solved directly to obtain

$$\delta_i = \frac{1}{T} \log \frac{S_i}{E_P[s_i(x)]} \tag{2.55}$$

This reduces to the generalized iterative scaling (GIS) algorithm (Darroch and Ratcliff, 1972). This is true, for example, for multinomial distributions where exactly one of the features has value 1 while the rest is 0. If $t(x)$ is not constant, an effective way of solving for (2.54) is by Newton's method, which, with care, can be made to always converge (Berger et al., 1996). For a gentle introduction to IIS and proof of (2.54), see (Berger, 1997); for further details consult (Berger et al., 1996).

For both IIS and GIS, instead of updating every parameter at each iteration, we can update

a subset U of the parameters at a time, keeping the rest fixed. In this case, each δ_i for $i \in U$ is again updated with (2.54) or (2.55), but with $t(x) = \sum_{i \in U} s_i(x)$ instead. In the extreme case when $t(x) = T = 1$ for all x, GIS reduces to the iterative proportional fitting (IPF) algorithm (Deming and Stephan, 1940). IPF steps can be much larger than GIS or IIS steps, since $t(x) = 1$ can be much smaller. This might make IPF sensitive to noise in sampling. However there are cases in which sampling is not required and IPF is preferable to GIS or IIS (Jiroušek and Přeučil, 1995, Teh and Welling, 2001).

Maximum entropy modelling has been successfully applied to the unsupervised modelling of textures (Zhu et al., 1997) and word morphology (Della Pietra et al., 1997). However perhaps the most effective use of maximum entropy is for modelling complex conditional distributions $P(x|z)$ which can be efficiently computed for each z without sampling. Examples of these applications of maximum entropy are found in (Nigam et al., 1999, Berger et al., 1996, Lafferty et al., 2001a).

2.5.2 Products of Experts

A product of experts (PoE) is a way of combining multiple models together to produce a more complex model. Each model $P_i(y, x_i|\theta_i)$, called an expert, has its own set of hidden variables x_i and parameters θ_i. The product is defined as

$$P(y, x|\theta) = \frac{1}{Z} \prod_i P_i(y, x_i|\theta_i) \tag{2.56}$$

where $\theta = \{\theta_i\}$ are the parameters of all the models and $x = \{x_i\}$ are all the hidden variables. Hinton (2002) has more information on PoEs. Two examples of PoEs are restricted Boltzmann machines (RBM) (Smolensky, 1986) and products of HMMs (Brown and Hinton, 2001). Initial results have indicated that PoEs are viable for natural language processing (Brown and Hinton, 2001), hand-written digit recognition (Mayraz and Hinton, 2001), face recognition (Teh and Hinton, 2001) and reinforcement learning (Sallans and Hinton, 2001).

Another way of combining models is a mixture of the models. A mixture model is a localist representation, while a product model uses distributed representation. In a mixture, one

component is chosen at a time to model one observation. Hence each component has to model all aspects of an observation, specializing in a small region of space where it models well. Because a mixture model cannot be more sharply tuned than each of its individual components, in high-dimensional spaces, a mixture will either need exponentially many components, or will not be able to model observations well. On the other hand, in a product model, each expert can model only one feature of the observation, and many experts can cooperate to model the whole observation. As a result, PoEs are more efficient models in high dimensions.

Marginalizing out x from (2.56), we have

$$P(y|\theta) = \sum_x \frac{1}{Z} \prod_i P_i(y, x_i|\theta_i) = \frac{1}{Z} \prod_i \sum_{x_i} P_i(y, x_i|\theta_i) = \frac{1}{Z} \prod_i P_i(y|\theta_i)$$
$$= \frac{1}{Z} \exp\left(\sum_i \log P_i(y|\theta_i) \right) \quad (2.57)$$

A PoE can be viewed as a special case of a maximum entropy distribution, where each feature comes from a differentiable parameterized family – the log likelihood of a simpler distribution. A maximum entropy distribution can also be thought of as a simple case of a PoE where each expert has a fixed shape and one parameter to determine how sharp the expert distribution is. In maximum entropy modelling, the features are used in a "black box" manner – they can be evaluated but are otherwise fixed and unknown. As a result, greedy search is needed to find a small set of features that gives a reasonably good performance in terms of likelihood. This may not be the optimal way of finding features. If the features involved come from a differentiable, parameterized family, then gradient ascent on likelihood is potentially a much more efficient method of finding useful features. This is the approach taken in PoEs.

Dividing (2.56) by (2.57), we have

$$P(x|y, \theta) = \prod_i P_i(x_i|y, \theta_i) \quad (2.58)$$

The hidden variables of each expert are conditionally independent from those of other experts. On the other hand, from (2.56), we see that the hidden variables are marginally dependent since they interact through the visible variables y. As a result inference in a PoE is straightforward if each expert is simple, but generating a sample is much more troublesome, and will have to

use Gibbs sampling with some convergence speeding method like simulated annealing. This situation is completely reversed from that of a causal model with a single layer of independent hidden units x_i and one layer of visible units (e.g. a two layer sigmoid belief network or factor analysis). In a causal model, the hidden units are marginally independent, while they are conditionally dependent given observations in the visible layer. Hence in causal models it is easy to generate a sample from the distribution, but hard to perform inference. Since inference is much more useful and common in practice than the ability to generate samples, it might be more desirable to use PoEs.

Just as for maximum entropy models, maximizing the log likelihood is not suitable for training a PoE with a fixed number of experts, because of the computational cost of evaluating or sampling from the partition function. Hinton (2002) proposed optimizing another function, coined contrastive divergence (CD), to make learning more efficient. Starting from the empirical distribution $Q^0(y) = \widetilde{P}(y)$, consider using Gibbs sampling to sample from the equilibrium distribution $Q^\infty(y) = P(y|\theta)$. Gibbs sampling proceeds as follows: fixing the visible units y, first sample from x_i for each i; then given the x's, sample from y. Let $Q^n(y)$ be the distribution obtained after n Gibbs iterations starting from $Q^0(y)$

$$Q^n(y) = \int Q^{n-1}(y')P(x'|y',\theta)P(y|x',\theta)\,dy'dx' \tag{2.59}$$

Then $Q^n(y) \to Q^\infty(y)$ as $n \to \infty$. Contrastive divergence is defined as

$$CD = KL(Q^0(y)\|Q^\infty(y)) - KL(Q^1(y)\|Q^\infty(y)) \tag{2.60}$$

Note that CD is always nonnegative and is exactly zero when $Q^0 = Q^1 = Q^\infty$, that is, the model has fit the data perfectly. Hence CD is a reasonable function to minimize. Differentiating (2.60) with respect to θ_i, we have

$$\frac{\partial CD}{\partial \theta_i} = -E_{Q^0}\left[\frac{\partial \log P_i(y|\theta_i)}{\partial \theta_i}\right] + E_{Q^1}\left[\frac{\partial \log P_i(y|\theta_i)}{\partial \theta_i}\right] + \int dy\, \frac{\partial CD}{\partial Q^1(y)}\frac{\partial Q^1(y)}{\partial \theta_i} \tag{2.61}$$

The first two terms of (2.61) can easily be estimated using samples from Q^0 and Q^1. The third term is much more troublesome. However, Hinton (2002) showed experimentally that the third

term can be safely ignored, leaving only the first two terms to estimate the gradient with respect to θ_i.

A problem with using CD learning is that (2.60) can be made small by having Gibbs sampling not mix properly, so that $Q^1 \approx Q^0$. This allows models trained with CD learning to be good at reconstructing the data, even when they are not modelling the data well in terms of high likelihood. To facilitate mixing, various methods can be used, for example, weight decay, and adding regularizers that encourage the reconstruction to be different from the data. Nevertheless, it has been shown that a RBM trained with CD learning is competitive, in the sense of high log likelihood, with a RBM trained with the standard Boltzmann machine (BM) learning rule[12]. But this is achieved with much less computational resources than BM learning, hence it is better to use CD learning for most practical purposes.

Another possibility for optimizing the parameters is by using iterated conditional modes (ICM) (Besag, 1983). Index the visible variables by j, so that $y = \{y_j\}$. Let $y_{\setminus j}$ be all visible variables except j and define the pseudo-likelihood as

$$PL = \sum_j E_{Q^0} \left[\log P(y_j | y_{\setminus j}, \theta) \right] \tag{2.62}$$

The pseudo-likelihood is an approximation to the log likelihood where we try to reconstruct each y_j from the other visible units. If each y_j is a discrete random variable, the pseudo-likelihood can be easily optimized because the partition function of $P(y_j | y_{\setminus j}, \theta)$ can be exactly computed. ICM is simply gradient ascent in PL. The derivative with respect to θ_i is

$$\frac{\partial PL}{\partial \theta_i} = \sum_j E_{Q^0(y) P(y'_j | y_{\setminus j}, \theta_i)} \left[\frac{\partial \log P_i(y_j | y_{\setminus j}, \theta_i)}{\partial \theta_i} - \frac{\partial \log P_i(y'_j | y_{\setminus j}, \theta_i)}{\partial \theta_i} \right] \tag{2.63}$$

Define $Q^1_{PL}(y) = \frac{1}{J} \sum_j Q^0(y_{\setminus j}) P(y_j | y_{\setminus j}, \theta)$, where J is the number of visible variables. Now using $\log P_i(y_j | y_{\setminus j}, \theta_i) = \log P_i(y | \theta_i) - \log P_i(y_{\setminus j} | \theta_i)$ we can show that (2.63) reduces to

$$\frac{\partial PL}{\partial \theta_i} = J \left(E_{Q^0} \left[\frac{\partial \log P_i(y | \theta_i)}{\partial \theta_i} \right] - E_{Q^1_{PL}} \left[\frac{\partial \log P_i(y | \theta_i)}{\partial \theta_i} \right] \right) \tag{2.64}$$

so ICM reduces to CD learning with a particular Gibbs sampling procedure, i.e. one where at each iteration j is chosen randomly and y_j is sampled from the conditional distribution

[12] Personal communication from S. Osindero.

$P(y_j|y_{\setminus j})$. The advantage of using ICM as opposed to using normal CD learning is that ICM is exactly maximizing the pseudo-likelihood so is guaranteed to converge. However, ICM is often more computationally intensive than CD learning. For example, in the case of RBM, each iteration of ICM requires $O(I^2J)$ operations, while each iteration of normal CD learning requires $O(IJ)$ operations, where I and J are the number of experts and visible variables respectively. In practice, we find that CD learning converges, hence there is no need to use the more expensive ICM.

2.6 Discussion

In this section we have covered the major techniques for inference and learning in graphical models, with particular emphasis on the approximate methods. In the rest of this thesis we will describe our contributions to these techniques.

Chapter 3

Rate-coded Restricted Boltzmann Machines for Face Recognition

In this chapter we explore a simple extension to restricted Boltzmann machines (RBMs) that allows them to model *discretized* continuous-valued random variables. This is achieved by *replicating* a single unit multiple times, and using the number of active replicas to represent the value of the random variable. These are called *rate-coded restricted Boltzmann machines* (RBMrate). We will show that RBMrate can be successfully applied to the tasks of facial modelling and recognition. However the fact that RBMrate can only model discretized and bounded data is unsatisfactory and leaves room for improvement. These problems with RBMrate serve as lessons learned and motivation for the energy-based models of the next chapter.

3.1 Products of Experts in Continuous-valued Domains

The first successful products of experts (PoEs), namely restricted Boltzmann machines and products of hidden Markov models, are all defined over essentially discrete domains (Mayraz and Hinton, 2001, Brown and Hinton, 2001, Sallans and Hinton, 2001). Hinton (2002) described a PoE where each expert is a "unigauss" model – it is a mixture between an axis-aligned

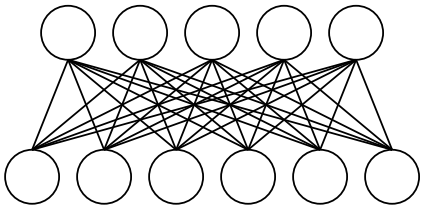


Figure 3.1: A restricted Boltzmann machine.

Gaussian and a uniform distribution[1]. It was shown that the product of unigauss models works well on some simple examples. However we found that it does not work as well on the problems we studied in this thesis, for example modelling images of faces and natural scenes. Extensions to mixtures between a factor analyzer and a uniform distribution ("unifact"), as well as mixtures between two factor analyzers have been unsuccessful as well.

In this and the next chapter we describe new classes of PoE that work well for the problems we are interested in. Rate-coded restricted Boltzmann machines (RBMrate) are described in this chapter, while in chapter 4 energy-based models (EBMs) are introduced.

3.2 Restricted Boltzmann Machines

A Restricted Boltzmann machine (RBM) is a Boltzmann machine with a layer of visible units v_i and a single layer of hidden units h_j with no hidden-to-hidden nor visible-to-visible connections (Smolensky, 1986, Hinton and Sejnowski, 1986). See figure 3.1. Let i and j be indices over visible and hidden units respectively. The distribution over $v = \{v_i\}$ and $h = \{h_j\}$ is given by

$$P(v, h) = \frac{1}{Z} e^{\sum_{i,j} W_{ij} v_i h_j} \tag{3.1}$$

where W_{ij} is the weight connecting v_i and h_j and Z is the normalizing constant. We have absorbed the biases over the visible units into the weights by having the activation of one of

[1]Technically a unigauss is an improper distribution due to the uniform distribution. There are two ways to correct this – we can either have an additional broad Gaussian expert so that the overall PoE becomes proper, or define the uniform distribution to have bounded support. The support should however be broad enough so that we can assume it is completely uniform in the region of space we are interested in.

the hidden units, say h_1, be fixed at 1. Similarly for the hidden unit biases. Pulling the sum over j out of the exponential,

$$P(v,h) = \frac{1}{Z}\prod_j e^{\sum_i W_{ij}v_i h_j} \tag{3.2}$$

we see that an RBM is a PoE with each hidden unit corresponding to an expert. As for any PoE, given observations on v the posterior over h is a factorized distribution with

$$P(h_j = 1|v) \;=\; \frac{1}{1+\exp(-\sum_i W_{ij}v_i)} \tag{3.3}$$

Hence inference in an RBM is much easier than in a general Boltzmann machine or in a causal belief network with one hidden layer. Conversely, the hidden units of an RBM are marginally dependent so an RBM may easily learn population codes in which units are highly correlated. It is hard to do this in causal belief networks with one hidden layer because the generative model of a causal belief net assumes marginal independence.

An RBM can be trained using the standard Boltzmann machine learning algorithm which follows a noisy but unbiased estimate of the gradient of the data log likelihood (Hinton and Sejnowski, 1986). However, just as for other PoEs, this requires samples from the prior distribution $P(v,h)$. MCMC sampling from the prior can take a long time to approach equilibrium and the sampling noise in the resulting estimate can obscure the gradient. As a result gradient ascent in the log likelihood of the data is not an effective learning algorithm.

We resort to using contrastive divergence to train RBMs instead. The learning rule is

$$\Delta W_{ij} \;\propto\; E_{P^0}\left[v_i h_j\right] - E_{P^1}\left[v_i h_j\right] \tag{3.4}$$

where

$$P^0(v,h) = \widetilde{P}(v)P(h|v) \tag{3.5}$$

is the "completed" data distribution, and P^1 is the distribution after running one step of MCMC sampling starting from P^0. Since the visible units are conditionally independent given the hidden activities and vice versa, we use Gibbs sampling where we alternate between updating all the hidden units in parallel, and updating all the visible units in parallel. Figure 3.2 illustrates

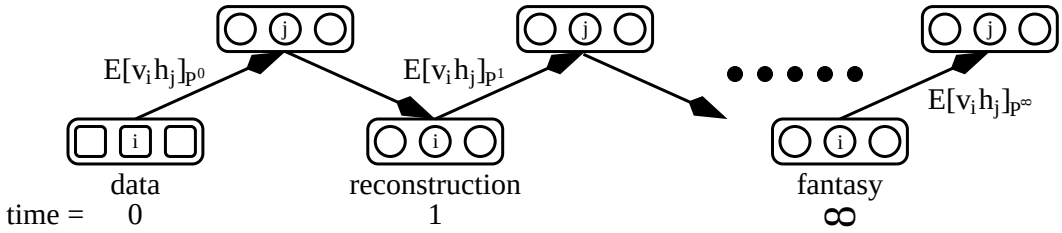


Figure 3.2: Alternating Gibbs sampling and the terms in the learning rules of a PoE.

this process. Since only one step of the MCMC sampling is performed, the noise in the estimate is significantly lower than if we try to sample from the prior using many steps. However this reduction in noise comes at the expense of introducing bias to the learned weights.

3.3 Rate-coded Restricted Boltzmann Machines

For continuous-valued data, the binary units of an RBM are far from ideal. Here we describe a simple way to increase the representational power without changing the inference and learning procedures of the RBM.

The idea is to "replicate" the units of the RBM multiple times, and use the number of active replicas of each unit to represent a *discretized* continuous value. Imagine that each visible unit, v_i, has n_i replicas which all have identical weights to the hidden units. So far as the hidden units are concerned, it makes no difference which particular replicas are turned on: it is only the number of active replicas that matters. So the group of replicas as a whole can now represent $n_i + 1$ different discretized values (i.e. $0, 1, 2, \ldots, n_i$). When sampling from the visible units given the hidden activities, all the replicas can share the computation of the probability, p_i, of turning on, and then we can select n replicas to be on with probability $\binom{n_i}{n} {p_i}^n (1-p_i)^{n_i-n}$. This is simply a binomial distribution with n_i trials, each with probability p_i being on. The same trick can be used for the hidden units h_j, where we replicate it n_j times. In summary, we have shown that both inference and sampling from an RBM can easily accommodate replicas. We can also easily show that learning with (3.4) is unaffected except that we replace v_i and h_j with the corresponding number of active replicas.

To be more precise, suppose that the continuous values we deal with are bounded between 0 and 1. We can model these values using the proportion of active replicas in each group. In particular define the aggregate random variable $\hat{v}_i = n/n_i$ where n is the number of active replicas corresponding to v_i, and similar for $\hat{h}_j$. We also rescale the weights by $\hat{W}_{ij} = W_{ij} n_i n_j$ to keep the distribution (3.1) the same. The distribution over the aggregate variables induced by the RBM model (3.1) is then given by

$$P(\hat{v}, \hat{h}) = \frac{1}{Z} \prod_{i,j} \binom{n_i}{\hat{v}_i n_i} \binom{n_j}{\hat{h}_j n_j} e^{\sum_{i,j} \hat{W}_{ij} \hat{v}_i \hat{h}_j} \tag{3.6}$$

This can be seen as a binomial extension to Boltzmann machines (the non-RBM extension is similar). In the following we will be dealing only with this model and not the original binary RBM. Hence when we refer to a unit we will mean instead the aggregate variable associated with the corresponding group of replicas. Similarly we will use v_i,h_j and W_{ij} instead of $\hat{v}_i$, $\hat{h}_j$ and $\hat{W}_{ij}$.

If each unit in a RBM represents a single neuron in the brain, the replica trick can be seen as a way of simulating the neuron over a time interval in which it may produce multiple spikes that constitute a rate-code. For this reason we call the model *rate-coded restricted Boltzmann machines* (RBMrate).

RBMrate is a simple way to model discretized continuous values between 0 and 1. We can render the discretization finer by increasing the number of replicas in each group. Unfortunately, this also means that the conditional distribution of an aggregate variable, say $\hat{v}_i$, given its neighbours $\hat{h}_j$ tends to a delta function centred at its mean p_i (since the variance of an aggregate variable, say $\hat{v}_i$ given $\hat{h}$ is $p_i(1 - p_i)/N$). Hence there is a trade off here between finer discretizations and more complex modelling capacity.

Nevertheless, we will show in the next two sections that it can still be successfully applied to the tasks of facial modelling and recognition. In the next chapter we will describe a generalization of PoEs that will naturally accommodate continuous values without discretization, bounding the continuous values to a certain range, or trade off between discretization resolution and model complexity.

3.4 RBMrate for Facial Modelling

As a test of concept, we applied RBMrate to the modelling of face images. We trained the RBMrate on 800 images of faces from the FERET database (Phillips et al., 1997). The images include the whole head, parts of the shoulder, and background. Instead of working with whole images, which contain much irrelevant information, we worked with normalized face images. The normalization is graphically depicted in figure 3.3 and involves five stages:

a. Original image.

b. Locate the centres of eyes by hand.

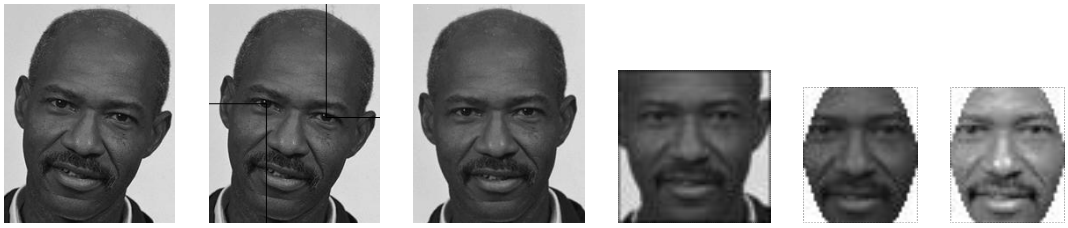
c. Rotate and translate the image so that the eyes are located at the preset locations.

d. Crop the image and subsample at 56×56 pixels.

e. Mask out all of the background and some of the face (with a fixed mask), leaving 1768 pixels in an oval shape.

f. Equalize the intensity histogram within the oval as follows: first compile the histogram of pixel intensities over all cropped images, then separately for each image we ordered the pixels according to their intensities and altered the intensities so that the intensity histogram matches that of the overall histogram.

Figure 3.4 shows some examples of the processed face images. Masking out all of the background inevitably looses the contour of the face which contains much useful information. The histogram equalization step removes most lighting effects, but it also removes some relevant information like the skin tone. In the ideal case this information should be modelled, so that if needed it can be used in further discriminative tasks like identity or expression recognition. Unfortunately, we, as well as other face modelling researchers, were not able to extract this information accurately while at the same time removing the unwanted pose and lighting information. However recent developments in image segmentation, for example Malik et al. (2001), have the potential to do this accurately.



(a) (b) (c) (d) (e) (f)

Figure 3.3: Normalizing the face images in the database.

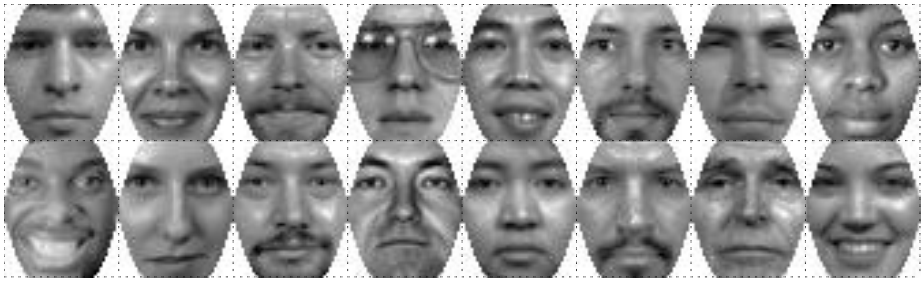


Figure 3.4: Some examples of processed faces.

The RBMrate model we applied has 100 hidden units, each of which has 100 replicas, while each of the 1768 visible units has 10 replicas. The model was trained with contrastive divergence with parallel Gibbs sweeps for 1000 iterations through mini-batches of size 100. We also made two further approximations: we replaced the expectation of $v_i h_j$ in (3.4) by the product of their expectations and we used the expected value of v_i when computing the probability of activation of the hidden units. However, we continued to use the stochastically chosen (discretized) firing rates of the hidden units when computing the one-step reconstructions of the data, so the hidden activities cannot transmit an unbounded amount of information from the data to the reconstruction. We have found empirically that replacing v_i with its expected value does not degrade performance and makes the computation more efficient. Replacing h_j with its expected value in (3.4) decreases the noise in the estimate of the contrastive divergence learning rule.

Some of the learned weights are shown in figure 3.5. All the units encode global features,

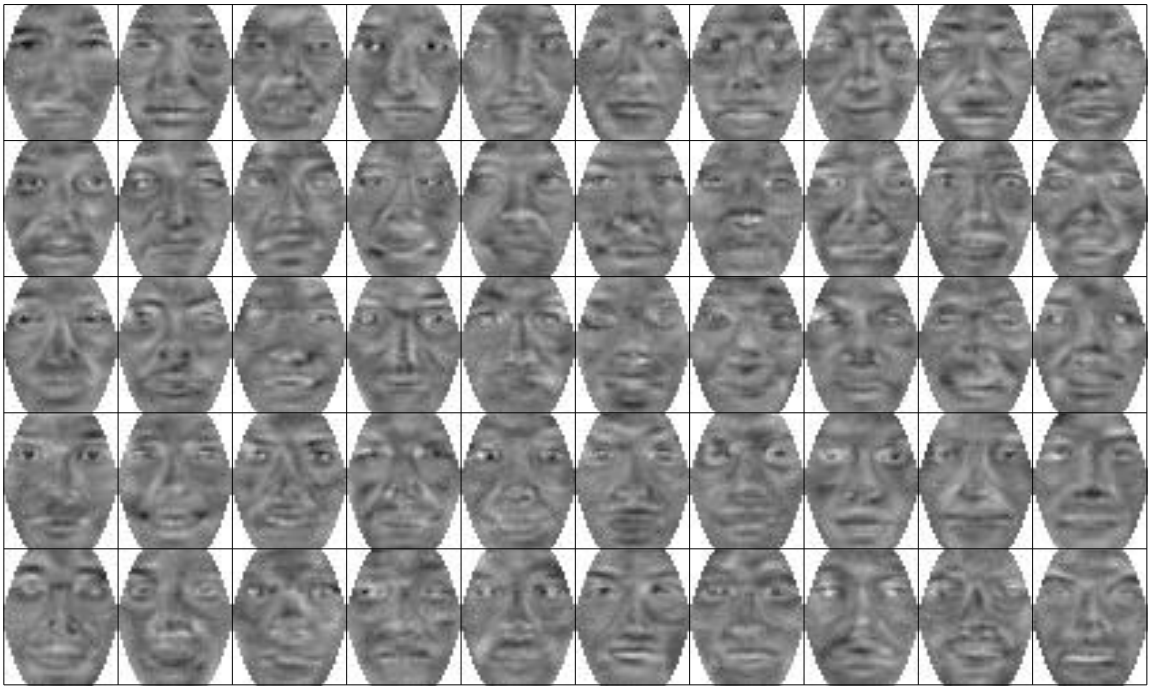


Figure 3.5: The weights learned by RBMrate. Each image is shows the weights adjacent to one hidden unit (called the receptive field).

probably because the image normalization ensures that there are strong long range correlations in pixel intensities. Next we compared face images with the model's reconstructions in figure 3.6. Given an original face image, the mean activities of the hidden units are inferred. We then reconstruct the image from the hidden activities of the model. The reconstructions are generally good enough. Notice however that glasses are not reconstructed well, since they do not occur very often in the dataset.

Inspired by (Lee and Seung, 1999), we tried to enforce local features by restricting the weights to be non-negative. This is achieved by resetting negative weights to zero after each weight update. Figure 3.7 shows some of the hidden receptive fields learned. Except for the 6 features on the left, all other features are local and code for features like mouths, eye brows, noses and cheeks. The 6 features on the left are much more global and clearly capture effects on the face when the lighting direction is changed.

Figure 3.6: Reconstructions of faces using the model. In each cell, the left image is the original, while the right one is the reconstruction after one Gibbs sampling iteration.

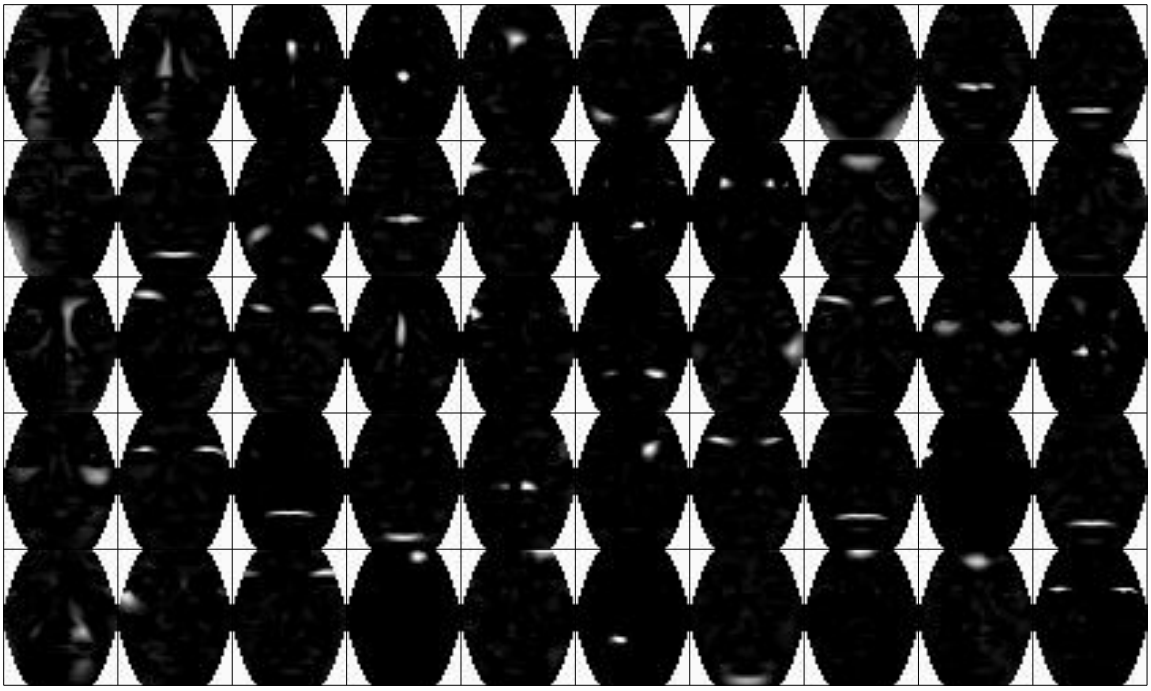


Figure 3.7: The weights learned by RBMrate when they are restricted to be non-negative.

The inferred hidden activities of the model can be understood as a representation of the original face image. This representation can be computed efficiently from the image since the hidden units of a PoE are conditionally independent. The representation also efficiently summarizes the original image (with 1768 pixels) using only 100 hidden unit activities, without much loss of information, as can be seen in figure 3.6. This facial representation can be used for further processing, for example, for recognizing the expression or identity of the individual in the image. We will apply a RBMrate model to identity recognition in the next section. Unfortunately, the representation of faces in this section cannot be fruitfully applied to face (identity) recognition because it does not take into account the fact that certain variations across faces are important for face recognition, while some variations are not. We will instead use a more suitable RBMrate model for pairs of images.

3.5 RBMrate for Face Recognition

Encouraged by the results in the previous section, we applied RBMrate to the task of face recognition. Face recognition is a difficult task since the number of individuals is typically very large and the test and training images of an individual differ in expression, pose, lighting and the date on which they were taken. In addition to being an important application for security or biometric verification, face recognition allows us to evaluate different kinds of algorithm for learning to recognize or compare objects, since it requires accurate representation of fine discriminative features in the presence of relatively large within-individual variations. This is made more difficult when, as istypical, there are very few exemplars of each individual.

At the same time, because of the large number of individuals typical in face recognition applications, the facial models used must have efficient algorithms to infer the activities of the facial features. This need for efficiency rules out many powerful models at the disposal of the researcher, and as a result most successful applications of face recognition employ simple models like principal components analysis and independent components analysis, where the facial features are simply linear combinations of the pixel intensities.

This combination of both the need to have flexible and powerful facial features and the need to have efficient inference makes PoEs ideal for face recognition. We will explore this application using RBMrate models in this section.

3.5.1 The FERET Database

The version of the FERET database (Phillips et al., 1997) we used contains 1002 frontal face images of 429 individuals taken over a period of a few years. To remove information irrelevant to face recognition, each image was normalized as in the previous section.

Of the 1002 images in the database, 818 are used both as the gallery and the training set[2]. The training set for the face-pair network consists of all pairs of faces from the gallery belonging to the same individual. There are 500 distinct such pairs, creating a training set of 1000 face pairs for RBMrate. To evaluate the strengths and weaknesses of RBMrate versus the other face recognition methods, we divided the remaining 184 images into 4 disjoint sets, each testing for a distinct condition. The test sets are:

- The **Δexpression** set contains 110 images of different individuals. These individuals all have another image in the training set that was taken under the same lighting conditions at the same time but with a different expression. The training set also includes a further 244 pairs of images that differ only in expression.

- The **Δdays** test set contains 40 images, two from each of 20 individuals. Each of these individuals has two images from the same session in the training set and two images taken in a session 4 days later or earlier in the test set. This test set evaluates the face recognition methods under changes in lighting conditions and slight variations in appearance (e.g. hair styles, make-ups, beard growth or shaved). A further 28 individuals were

[2]The *gallery* is a set of images of individuals whose identities are known. Given a *test* image, it is compared against the images in the gallery and the individual in the most similar one (based on some *similarity measure*) is identified as the person in the test image. The similarity measure is typically computed using a model (for example, in eigenfaces (Turk and Pentland, 1991) the model used is principal components analysis) which is trained using a *training set*. Here we used the same images in the gallery and training set, although they could differ (or even be disjoint). The test sets are disjoint from both the training set and the gallery.

photographed in a similar fashion and all 112 of these images are in the training set.

- The **Δmonths** test set is just like the Δdays test set except that the time between sessions was at least three months. This test set evaluates major changes in the set-up (e.g. different rooms, cameras etc), lighting conditions, and appearance. It is most interesting as it reflects conditions in real life applications of face recognition. This set contains 20 images of 10 individuals. A further 36 images of 9 more individuals were included in the training set.

- The **Δglasses** test set contains 14 images of 7 different individuals. Each of these individuals has two images in the training set that were taken in another session on the same day. The training and test pairs for an individual differ in that one pair has glasses and the other does not. This test set evaluates a major change in appearance (i.e. with or without glasses) while controlling for changes in lighting conditions or set-up. The training set includes a further 24 images, half with glasses and half without, from 6 more individuals.

3.5.2 Popular Face Recognition Methods

We compared RBMrate with four popular face recognition methods:

- The first and simplest is **correlation**, which returns the similarity score as the angle between the two images represented as vectors of pixel intensities. Correlation performed better than using the Euclidean distance as a score.

- The second method is **eigenfaces** (Turk and Pentland, 1991), which projects the images onto the principal component subspaces, and returns the similarity score as the angle between the projected images. The principal components are determined from the training set. The first few principal components could be removed if they were manually determined to be coding for features which are invariant for face recognition. This tends to improve recognition performance. In our case, we omitted the first principal component as it encodes for changes in lighting conditions, and used the next 199 components

instead. Omitting the first principal component improved recognition performance on all test sets except for Δexpression (which is reasonable since Δexpression images are taken under the same lighting conditions, but we know we do not want to use that information for face recognition).

- The third method is **fisherfaces** (Belmumeur et al., 1996). This method is like eigenfaces, except instead of projecting the images onto the subspace of the principal components (thus maximizing the variance among the projected images) fisherfaces projects the images onto a subspace which both maximizes the *between individual* variances and minimizes the *within individual* variances in the training set. The intuition is that we want the clusters corresponding to different classes (individuals) to be as well separated as possible to improve discrimination. We used a subspace of dimension 200 for this projection.

- The final method, which we shall call ***δ*ppca**, is proposed by Moghaddam *et al* (Moghaddam et al., 1998). This method models differences between images of the same individual as a probabilistic principal components analysis (PPCA) (Moghaddam and Pentland, 1997, Tipping and Bishop, 1997), and differences between images of different individuals as another PPCA. Then given a difference of two images, it returns as the similarity score the likelihood ratio of the difference image under the two PPCA models. It was the best performing algorithm in the September 1996 FERET test (Phillips et al., 1997). We used 10 and 30 dimensional PPCAs for the within-class and between-class models respectively. These are the same numbers used by Moghaddam et al. (1998) and gives the best results in our simulations.

3.5.3 A Face-pair RBMrate Model

A simple way to use RBMrate for face recognition is to train a single RBMrate to model faces as in section 3.4, and to identify a face by finding the gallery image that produces a hidden activity vector that is "most similar" to the one produced by the face. This is how eigenfaces

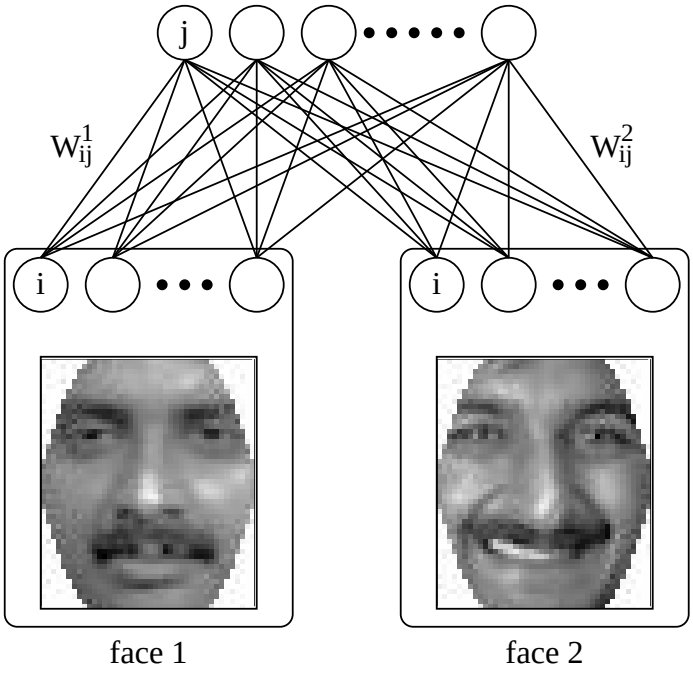


Figure 3.8: A RBMrate model for pairs of faces.

(Turk and Pentland, 1991) are used for recognition, but it does not work well because it does not take into account the fact that some of the variations across faces are important for recognition, but other types of variations are not. To correct this, we instead trained an RBMrate model on pairs of different images of the same individual, and then use this model of pairs to decide which gallery image is best paired with the test image.

This model is similar to that used in section 3.4. There are 100 hidden units, each with 100 replicas, while the visible units have 10 replicas each. The model is trained on pairs of different face images belonging to the same individual. Let the weights connecting hidden unit j to pixel i in the first and second images be W_{ij}^1 and W_{ij}^2 respectively (see figure 3.8). We did not find it necessary to use weight sharing, although it is possible in principle. Note however that the obvious way to share weights, i.e. $W_{ij}^1 = W_{ij}^2$, is not desirable since it means that hidden units cannot describe variations across the faces of a single individual (see later). Instead, weight sharing should be imposed on pairs of hidden units j_1 and j_2 by $W_{ij_1}^1 = W_{ij_2}^2$ and $W_{ij_1}^2 = W_{ij_2}^1$.

To preserve symmetry, each pair of images of the same individual (v_1, v_2) in the training

set also has a reversed pair (v_2, v_1) in the set. We trained the model on 1000 image pairs (500 distinct pairs) for 2000 iterations in batches of 100, with a learning rate of $.0025$ for the weights, a learning rate of $.005$ for the biases, and a momentum of 0.95[3]. These parameters were chosen so that the RBMrate can learn quickly without diverging (the learning can diverge simply because the step sizes are too large).

Given a face pair $v = (v^1, v^2)$, we define the goodness of fit to the model to be the negative free energy of v under the model, i.e.

$$G(v^1, v^2) = G(v) = \sum_h P(h|v) \left(\sum_{k=1,2} \sum_{i,j} W_{ij}^k v_i^k h_j - \log P(h|v) \right) \tag{3.7}$$

Note that $G(v^1, v^2) \neq G(v^2, v^1)$ in general. However the two quantities will be approximately equal as the training set is symmetric with respect to switching the faces in each pair.

To account for fact that certain face images are intrinsically easier to model than others, we introduce a balanced similarity measure $F(v^1, v^2)$ given by

$$F(v^1, v^2) = G(v^1, v^2) + G(v^2, v^1) - G(v^1, v^1) - G(v^2, v^2) \tag{3.8}$$

To determine the identity of an individual, we take the test image f and find the gallery image g that maximizes $F(f, g)$.

In the presence of many face images each with many pixels, the efficiency of face recognition methods is an important concern. Our face-pair network shares an advantage of methods like eigenfaces and fisherfaces in that comparisons of test and gallery images can be made in the low-dimensional feature space rather than the high-dimensional space of pixel intensities. Here we show how to do this comparison.

Given an image pair $v = (v^1, v^2)$, if we let $I_j^k = \sum_i W_{ij}^k v_i^k$ be the total input into hidden unit h_j from image v^k, then the mean of h_j is given by $p_j = 1/(1 + \exp(-\frac{I_j^1 + I_j^2}{n_j}))$. The

[3]The learning rates here are 10^3 larger than those reported in (Teh and Hinton, 2001) because we are using the proportion of active replicas $\hat{v}_i$ here, rather than the number of replicas.

goodness of fit to the model is, after some algebraic manipulation,

$$G(v^1, v^2) = \sum_{i,j,k} W_{ij}^k v_i^k p_j - \sum_j n_j \left(p_j \log p_j + (1 - p_j) \log(1 - p_j) \right) \tag{3.9}$$

$$= \sum_j n_j \log \left(1 + \exp \left(\frac{I_j^1 + I_j^2}{n_j} \right) \right) \tag{3.10}$$

Hence both the goodness of fit and the similarity between two images can be computed efficiently using only the total inputs into the hidden units I_j^k. Thus if we precompute and store these total inputs for the gallery images, and for each test image compute them once, we can compute the similarity between the test image and all the gallery images efficiently.

3.5.4 Comparative Results

Figure 3.9 shows the error rates of all five methods on the test sets. The results were averaged over 10 random partitions of the dataset to improve statistical significance. Correlation and eigenfaces perform poorly on Δexpression, probably because they do not attempt to ignore the within-individual variations, whereas the other methods do. All the models did very poorly on the Δmonths test set which is unfortunate as this is the test set that is most like real applications. RBMrate performed best on Δexpression, fisherfaces is best on Δdays and Δglasses, while eigenfaces is best on Δmonths. These results show that RBMrate is competitive with but does not perform better than other methods. To demonstrate the difficulty of the Δmonths test set, we have produced figure 3.10. The Δmonths test set is intrinsically difficult and is made even harder by the loss of contour and skin tone information.

3.5.5 Receptive Fields Learned by RBMrate

Figure 3.11 shows the weights of a few of the hidden units after training the RBMrate. All the units encode global features, perhaps because the image normalization ensures that there are strong long range correlations in pixel intensities that are easily captured by the RBMrate. The maximum size of the weights is 0.01765, with most weights having magnitudes smaller than 0.005. Note, however, that the hidden unit activations range from 0 to 100. Figure 3.11

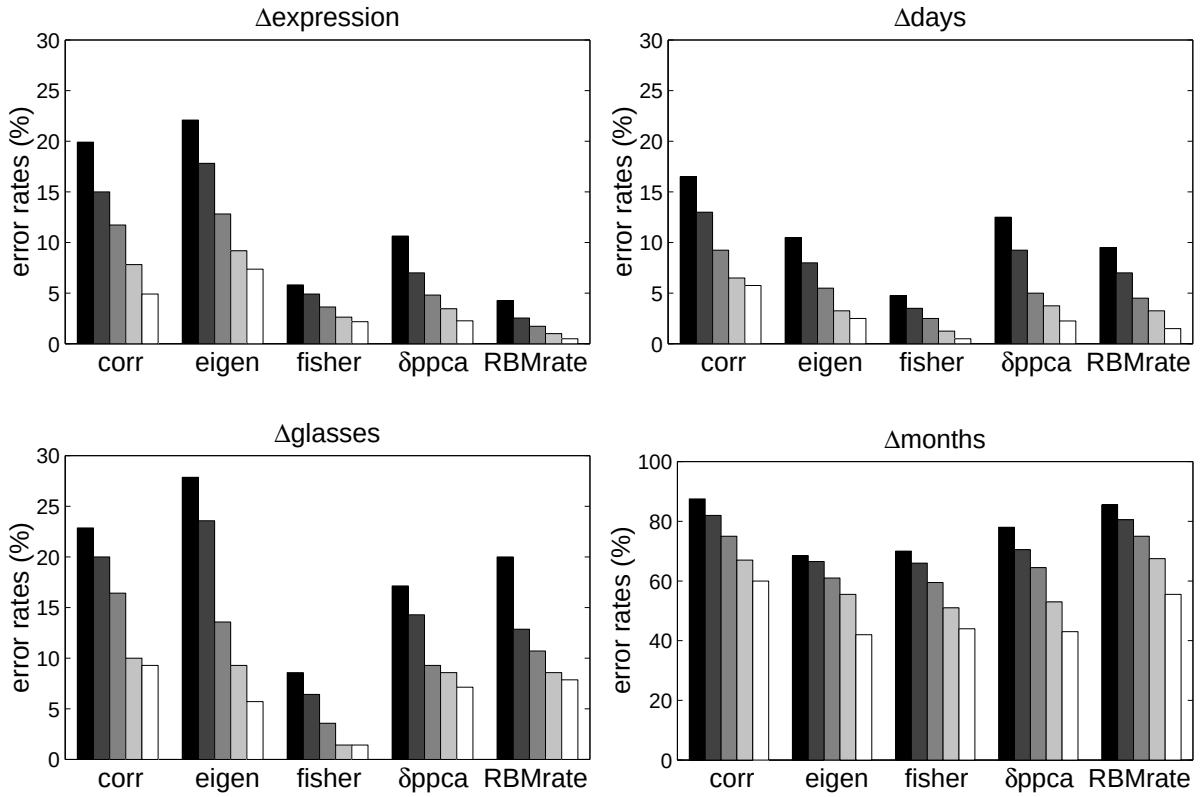


Figure 3.9: Error rates of all methods on all test sets. The bars in each group correspond, from left to right, to the rank-1, rank-2, rank-4, rank-8 and rank-16 error rates. The rank-n error rate is the percentage of test images where the n most similar gallery images are all incorrect.

Figure 3.10: On the left is a test image from Δmonths and on the right are the 8 most similar images returned by RBMrate. Most human observers cannot find the correct match to the test image.

shows that RBMrate has discovered some features which are fairly constant across images in the same class, and some features which can differ substantially within a class. For example, the top left feature may encode the presence of mustache in both faces, the feature below that may code for prominent right eyebrows in both faces, while the feature to the right may encode

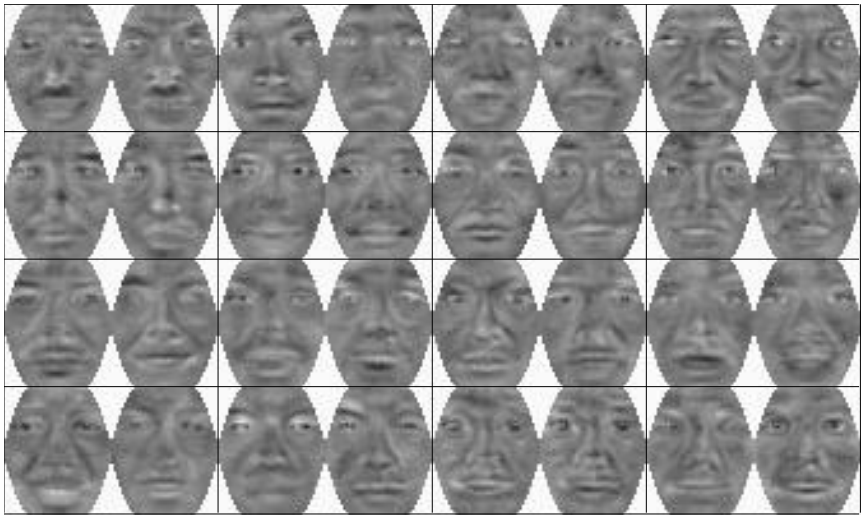


Figure 3.11: Example features learned by RBMrate. Each pair of receptive fields constitutes a feature.

the fact that the mouth shape may change across images of the same individual.

We can again let RBMrate learn local features by constraining the weights to be non-negative. Figure 3.12 shows some of the hidden receptive fields learned. Except for the four features on the left, all other features are local and code for features like mouth shape changes (third column) and eyes and cheeks (fourth column). The four features on the left are much more global and clearly capture the fact that the direction of the lighting can differ for two images of the same person. Constraining the weights to be non-negative strongly limits the representational power of RBMrate and makes it perform worse than all the other methods on all the test sets.

3.6 Discussion

We have introduced an extension to RBMs, called RBMrate, that models discretized continuous values. Each unit of the RBMrate models can be understood as representing the number of spikes of a neuron over a time interval.

We applied RBMrate models to face modelling and recognition, and showed that they pro-

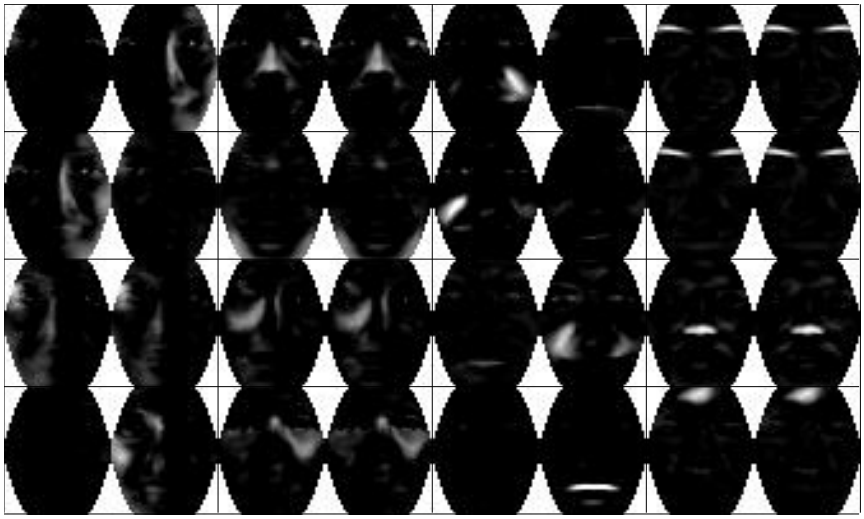


Figure 3.12: Example features learned by RBMratewith non-negative weight constraints.

duce efficient facial models and that they are comparable to popular methods for face recognition. However, unlike other face recognition methods based on linear models, there is plenty of room for further development, for example using prior knowledge to constrain the weights or adding additional layers of hidden units to model the correlations of hidden activities as in (Mayraz and Hinton, 2001). These improvements should translate into improvements in the rate of recognition.

Acknowledgements and Remarks

We thank Jonathon Phillips for graciously providing us with the FERET database. An earlier version of this chapter first appeared in the Neural Information Processing Systems conference held in Denver, Colorado in December 2000 (Teh and Hinton, 2001). We thank the Gatsby Charitable Foundation for funding this project.

Chapter 4

Energy-Based Models for Sparse Overcomplete Representations

In this chapter we introduce a more natural extension of PoEs to continuous-valued domains. In these *energy-based models* (EBMs), each expert is simply some (potentially non-linear) function of the data which contributes an energy term. The overall probability distribution over the data space is defined as the Boltzmann distribution corresponding to the sum of the energies. We assume that the energies are such that the Boltzmann distribution is properly defined (i.e. the exponentiated negative energies is normalizable). If each expert is simply some non-linearly transformed output of a linear filter on the data, EBMs turn out to be a novel and interesting generalization of square noiseless independent components analysis (ICA). ICA is a popular linear but non-Gaussian model that is used for blind source separation and the extraction of features from sounds and images. The focus of this chapter is to describe this simpler case and its relationship to ICA. In particular, we show that this energy-based approach of ICA is equivalent to the established approaches of ICA in the square and noiseless case, but gives novel and interesting extensions of ICA to the over-complete case.

In section 4.1 we describe the three views of ICA – the causal generative view (Pearlmutter and Parra, 1996, MacKay, 1996, Cardoso, 1997), the information maximization view (Bell and Sejnowski, 1995, Shriki et al., 2002), and our new energy-based view. In sections 4.2 and

4.3 we describe the square noiseless ICA and over-complete extensions from both the causal generative and filtering perspectives in detail. In section 4.4 we describe our energy-based view of ICA and its relationship to the other views. We describe contrastive divergence learning of energy-based ICA in section 4.5, and experiments in sections 4.6 and 4.7, and close with some discussion in section 4.8.

4.1 Introduction

There have been two dominant ways of understanding ICA, one based on a bottom-up, filtering approach and the other based on a top-down, causal generative approach. In the information maximization view (Bell and Sejnowski, 1995, Shriki et al., 2002) the aim is to maximize the mutual information between the observations and the non-linearly transformed outputs of a set of linear filters. In the causal generative view (Pearlmutter and Parra, 1996, MacKay, 1996, Cardoso, 1997), on the other hand, the aim is to build a density model in which independent, non-Gaussian sources are linearly combined to produce the observations.

The main point of this chapter is to show that there is a third, "energy-based" view of ICA which combines a bottom-up, filtering approach with the goal of fitting a probability density to the observations. The parameters of an energy-based model specify a deterministic mapping from an observation vector $\mathbf{x}$ to a feature[1] vector and the feature vector determines a global energy, $E(\mathbf{x})$. The probability density of $\mathbf{x}$ is defined by:

$$p(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z} \tag{4.1}$$

where Z is a normalization factor - the integral of the numerator over all possible observation vectors. The energy-based view is interesting because it suggests novel and tractable ways of extending ICA to over-complete and multi-layer models.

The relationship between the three perspectives is depicted in figure 4.1. In general, they are quite different, but they all become equivalent for the "square" and noiseless case, that

[1]When discussing energy-based models, we use the term "feature" rather than "source" for reasons that will become clear when we discuss extensions to the over-complete case.

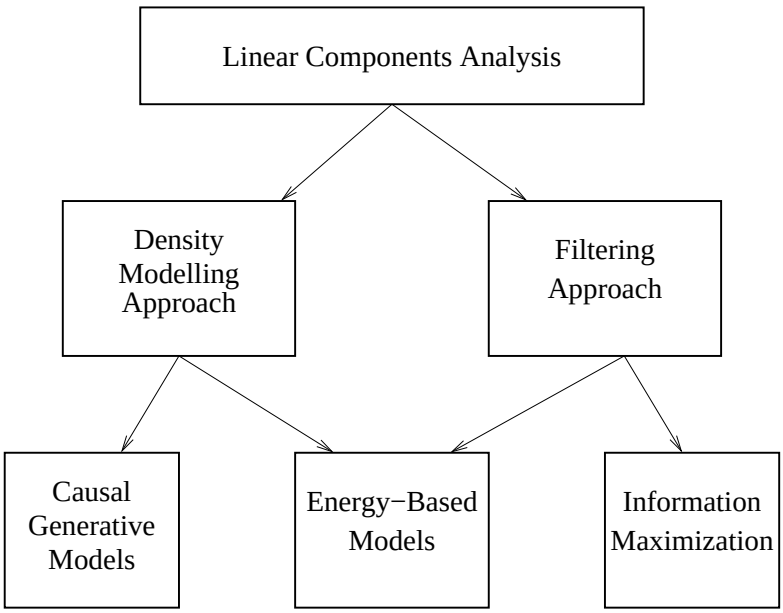


Figure 4.1: Different methods for non-Gaussian linear components analysis.

is, when the number of sources or features equals the number of observations and there is no observation noise. While complete representations have been applied successfully to a wide range of problems, researchers have recently argued for "over-complete" representations where there are more sources or features than observations. Apart from greater model flexibility, reported advantages range from improved robustness in the presence of noise (Simoncelli et al., 1992), more compact and more easily interpretable codes (Mallat and Zhang, 1993) to super-resolution (Chen et al., 1998).

The natural way to extend the causal generative view to the over-complete case is to retain the assumption that the sources are independent when the model is used to generate data and to accept the consequence that an observation vector creates a posterior distribution over a multiplicity of source vectors. In this posterior distribution, the sources are conditionally dependent due to the effect known as "explaining-away" and, in general, the distribution has the unfortunate property that it is computationally intractable.

The natural way to extend the information maximization view to over-complete representations is to retain both the simple, deterministic, feedforward filtering of observations and the

Models	Unconditional distribution over "source" vectors (before an observation)	Conditional distribution over "source" vectors (after an observation)
Square	Independent (by assumption)	Independent (because deterministic)
Causal Over−complete	Independent (by assumption)	Dependent (explaining−away)
Energy−based Over−complete	Dependent (rejecting−away)	Independent (because deterministic)

Figure 4.2: Independence properties of three types of models.

mutual information objective function (Shriki et al., 2002). However, because the manifold of possible filter outputs typically does not consist of the whole space (except in the square case), the equivalence with causal generative models breaks down.

When our energy-based view of ICA is made over-complete, it continues to be a proper density model and it retains the computationally convenient property that the features are a deterministic function of the observation vector. However, it abandons the marginal independence of the features (which is why we do not call them sources). A useful way of understanding the difference between energy-based density models and causal generative density models is to compare their independence properties. Figure 4.2 summarizes the similarities and differences.

The table reminds us that the different views are equivalent in the square case, and hence, in the absence of observations, the sources are marginally independent. Further, the posterior distribution over source vectors conditioned on an observation vector collapses to a point in the absence of noise, so the sources are trivially independent in the posterior distribution. In the causal generative approach this conditional independence of the sources is seen as a fortuitous consequence of using as many sources as observations and avoiding noise in the observations, and is not retained in the over-complete case. In the energy-based view, the conditional independence of the features is treated as a basic assumption that remains true even in the over-complete case. We can consider the energy contributed by the activity of a feature

of an energy-based model as the negative log probability of a one-dimensional, non-Gaussian distribution. However not all combinations of feature activities can occur because the lower-dimensional observation space only maps to a restricted manifold in the feature space.

The marginal dependence of the features in an over-complete, energy-based model can be understood by considering an illuminating but infinitely inefficient way of generating unbiased samples from the energy-based density model. First we sample the features independently from their "prior" distributions (the negative exponentials of their individual energy contributions) and then we reject cases in which the feature activities do not correspond to valid observation vectors. Since the features are over-complete, if the feature values are accepted, there is only one observation vector consistent with the feature values (assuming non-degeneracy). This process of "rejecting-away"creates dependencies among the activities of different features.

Because of the confusion between marginal and conditional independencies in the various models, we avoided using the term *independent* component analysis in figure 4.1. Rather we used *linear* component analysis instead, since there is always a linear relationship between the sources/features and the observations.

For some applications, such as unmixing sound sources, the causal generative view is clearly more appropriate than the energy-based view because we have a strong prior belief that the sources are marginally independent. In many other applications, however, the real aim is to model the probability density of the data, or to discover interpretable structure in the data, or to extract a representation that is more useful for controlling action than the raw data itself. In these applications, there is no *a priori* reason for preferring the causal generative view to the energy-based view that characterizes each observation vector by representing the degree to which it satisfies a set of learned features.

4.2 Square ICA

In this section we will briefly review the standard models for ICA. One of the first expositions on ICA (Comon, 1994) used the entropy of linearly transformed input vectors as a *contrast*

function to find statistically independent directions in input space. Indeed many, if not all, ICA algorithms ultimately reduce to optimizing some sort of "contrast function"; this overview will not try to mention them all. Rather we will focus on reviewing two general approaches to ICA, namely the causal generative approach (Pearlmutter and Parra, 1996, MacKay, 1996, Cardoso, 1997) and the information maximization approach (Bell and Sejnowski, 1995, Shriki et al., 2002). Subsequent sections will then compare these canonical approaches with our proposed energy-based approach, and in particular will explore the consequences of making the different models over-complete.

Consider a real valued input, denoted by $\mathbf{x}$, of dimensionality D, and an M-dimensional source or feature vector, denoted by $\mathbf{s}$. In this section we will consider the special case where the number of input dimensions is equal to that of the source or features, i.e. $D = M$.

4.2.1 The Causal Generative Approach

In the causal generative approach, the sources $\mathbf{s}$ are assumed to be independent, that is, the distribution $p(\mathbf{s})$ factors as

$$p(\mathbf{s}) = \prod_{i=1}^{M} p_i(s_i) \tag{4.2}$$

while the inputs are simply linear combinations of the sources. Moreover, we will assume for now that there is no noise on the inputs, i.e.

$$\mathbf{x} = A\mathbf{s} \tag{4.3}$$

where A is a square invertible matrix called the mixing matrix. Inverting this relationship we have

$$\mathbf{s} = W^T\mathbf{x} \qquad\qquad W^T = A^{-1} \tag{4.4}$$

where the inverse mixing matrix W^T will be called the filter matrix since each row acts as a linear filter of the inputs.

The aim is now to recover the statistically independent source signals $\mathbf{s}$ from the linearly mixed observations $\mathbf{x}$. This turns out to be possible only if the statistical properties of the

sources are non-Gaussian[2]. Thus, we shall assume that the probability distribution of the sources will be modelled by non-Gaussian prior distributions $p_i(s_i)$. Since the relation between sources and inputs is deterministic and one-to-one, we may view it as a change of coordinates. Deriving an expression for the probability distribution of the inputs $\mathbf{x}$ can therefore be accomplished by transforming expression (4.2) to $\mathbf{x}$-space, using the Jacobian of that transformation,

$$p(\mathbf{x}) = \left|\det \frac{\partial \mathbf{s}}{\partial \mathbf{x}}\right| \prod_{i=1}^{M} p_i(s_i(\mathbf{x})) = |\det W| \prod_{i=1}^{M} p_i(\mathbf{w}_i^T \mathbf{x}) \tag{4.5}$$

where $\mathbf{w}_i$ are the columns of W. Learning proceeds by averaging the log-likelihood for the above model over a data distribution[3] $p^0(\mathbf{x})$ and using the derivatives of it with respect to W for gradient ascent:

$$\frac{\partial E_{p^0}\left[\log p(\mathbf{x})\right]}{\partial w_{ij}} = E_{p^0}\left[\frac{\partial \log p_i(s_i)}{\partial s_i} x_j\right] - \left[W^{-T}\right]_{ij} \tag{4.6}$$

where w_{ij} is the j^{th} entry of $\mathbf{w}_i$ and $[W^{-T}]_{ij}$ is the ij^{th} entry of the matrix $W^{-T} = (W^T)^{-1}$.

4.2.2 The Information Maximization Approach

An alternative, more neurally plausible approach to ICA was put forward by Bell and Sejnowski (1995)[4]. In that paper it was assumed that a certain transformation was applied to the inputs,

$$y_i = f_i(\mathbf{w}_i^T \mathbf{x}) \qquad\qquad i = 1 \ldots M \tag{4.7}$$

with $f_i(\cdot)$ being a monotone squashing function such as a sigmoid and $\mathbf{w}_i$ a set of linear filters. It was then argued that maximizing the mutual information[5] between outputs $\mathbf{y}$ and inputs $\mathbf{x}$, which is equivalent to maximizing the *entropy* of $\mathbf{y}$ due to the deterministic relation (4.7),

[2]Technically at most one of the sources can be Gaussian.

[3]This data distribution is the underlying distribution from which our observed data is sampled. In practice, we replace this by the empirical distribution over the training set.

[4]In fact this information maximization approach to ICA was proposed first, followed by the causal generative approach. We presented the two approaches in reverse order here for more intuitive exposition.

[5]Note that this mutual information is measured with respect to the data distribution p^0.

would lead to independent components. This effect can be understood through the decomposition

$$H(\mathbf{y}) = \sum_{i=1}^{M} H_i(y_i) - I(y_1, \ldots, y_M) \tag{4.8}$$

with $H(\mathbf{y})$ the entropy of $\mathbf{y}$, $H_i(y_i)$ the individual entropies, and I the mutual information among y_i's. Maximizing the joint entropy thus involves maximizing the individual entropies of the y_i's and *minimizing* the mutual information between the y_i's, i.e. making the y_i's independent.

This approach can best be described as a filtering approach, since each y_i is just a squashed version of the filter outputs $s_i = \mathbf{w}_i^T \mathbf{x}$. This is in contrast with the causal generative approach where we instead think of $\mathbf{x}$ as being generated by $\mathbf{s}$ in a top-down manner.

4.2.3 Equivalence of the Two Approaches

For square representations the information maximization approach turns out to be *equivalent* to the causal generative one if we interpret $f_i(\cdot)$ to be the cumulative distribution function of $p_i(\cdot)$ (Pearlmutter and Parra, 1996, MacKay, 1996, Cardoso, 1997). This can be seen by observing that the entropy $H(\mathbf{y})$ can be written as a negative KL divergence using a change of variables as follows,

$$H(\mathbf{y}) = -\int d\mathbf{y}\; p^0(\mathbf{y}) \log p^0(\mathbf{y}) = -\int d\mathbf{x}\; p^0(\mathbf{x}) \log \frac{p^0(\mathbf{x})}{|\det J(\mathbf{x})|} \tag{4.9}$$

where $p^0(\mathbf{y}) = \frac{p^0(\mathbf{x})}{|\det J(\mathbf{x})|}$ and $J(\mathbf{x})$ is the Jacobian of the transformation between $\mathbf{y}$ and $\mathbf{x}$,

$$J_{ij}(\mathbf{x}) = \frac{\partial y_i(\mathbf{x})}{\partial x_j} \tag{4.10}$$

Using some basic algebra it can be shown that $|\det J(\mathbf{x})|$ is in fact exactly equal to (4.5), since f_i satisfies $f_i'(\cdot) = p_i(\cdot)$, being the cumulative function of p_i. Therefore maximizing the entropy in (4.9) is indeed equivalent to maximizing the log likelihood of the model (4.5). Note also that if the sources are *actually* distributed according to $p_i(\cdot)$ and mixed using A, then the transformation (4.7) maps the input variables to independent, uniformly distributed variables over the range of $f_i(\cdot)$, i.e. the interval $[0, 1]$ in case of a sigmoid. This geometric interpretation will be of help in section 4.3.2.

4.2.4 Square ICA with Input Noise

In the previous section we have shown that the causal generative approach, in the special case of a square mixing matrix and no noise, is equivalent to the information maximization approach. This equivalence will break down, however, when we consider a noise model for the inputs. In that case, there is no longer a deterministic relationship between the inputs and the sources. It is, however, still straightforward to write a probabilistic model for the joint distribution over sources and inputs,

$$p(\mathbf{x}, \mathbf{s}) = p(\mathbf{x}|\mathbf{s}) \prod_{i=1}^{M} p_i(s_i) \tag{4.11}$$

Unfortunately, even for isotropic Gaussian noise it is no longer true that given a mixing matrix A the optimal reconstruction of the sources is simply given by (4.4). Instead, one typically computes the maximum a posteriori (MAP) value (or the mean, depending on the objective) of the *posterior* distribution $p(\mathbf{s}|\mathbf{x})$.

4.3 Overcomplete Generalizations of ICA

The equivalence between the two approaches also breaks down when there are more sources than input dimensions, i.e. when we consider over-complete representations of the data. We will now review over-complete generalizations of ICA based on both approaches.

4.3.1 The Causal Generative Approach

Arguably the most natural way to extend the ICA framework to over-complete representations is through the causal generative approach. The corresponding directed graphical model is depicted in figure 4.3a. For noiseless inputs, finding the most probable state $\mathbf{s}$ corresponding to a particular input $\mathbf{x}$ now translates into the following optimization problem,

$$\mathbf{s}^{\mathbf{MAP}} = \underset{\mathbf{s}}{\operatorname{argmax}} \left\{ \sum_{i=1}^{M} \log p_i(s_i) \;\middle|\; \text{such that } \mathbf{x} = A\mathbf{s} \right\} \tag{4.12}$$

The above problem is typically hard and it can only be solved efficiently for certain choices of $p_i(\cdot)$. For instance Lewicki and Sejnowski (2000) argued that by choosing the priors to be

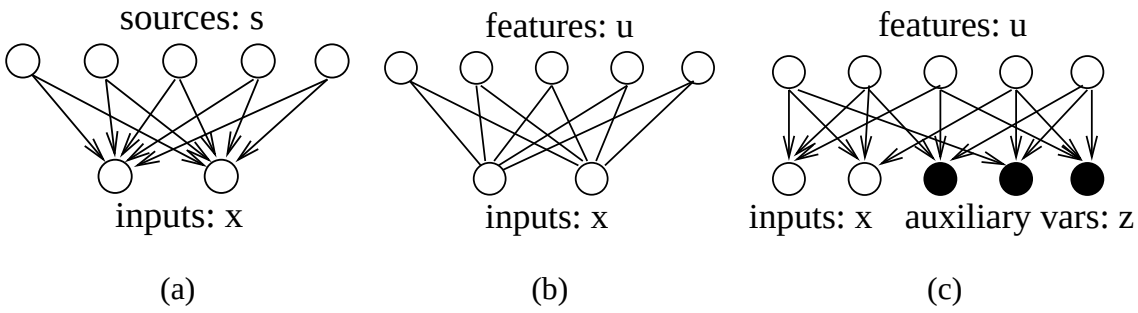


Figure 4.3: (a) Directed graphical model corresponding to the causal generative approach to ICA. (b) Undirected graphical model for an EBM. (c) Directed graphical model representation for an EBM with auxiliary variables clamped at 0.

Laplacian the problem can be mapped to a standard linear program.

One can "soften" this optimization problem by introducing a noise model for the inputs. For instance, using a spherical Gaussian noise model with noise variance σ^2 we find the following joint probability density distribution over sources and inputs,

$$p(\mathbf{x}, \mathbf{s}) = p(\mathbf{x}|\mathbf{s})\; p(\mathbf{s}) = \mathcal{N}_{\mathbf{x}}\left[A\mathbf{s}, \sigma^2 I\right] \;\prod_{i=1}^{M} p_i(s_i) \tag{4.13}$$

which leads to the following maximization problem to reconstruct the sources from the inputs,

$$\mathbf{s}^{\mathbf{MAP}} = \underset{\mathbf{s}}{\operatorname{argmax}}\left\{ -\frac{1}{2\sigma^2}|\mathbf{x} - A\mathbf{s}|^2 + \sum_{i=1}^{M} \log p_i(s_i) \right\} \tag{4.14}$$

Maximum likelihood learning for the above noisy model using the EM procedure involves averaging over the posterior distribution $p(\mathbf{s}|\mathbf{x})$. Unfortunately this inference problem is intractable in general and approximations are needed. In the literature one can find a whole range of approximate inference techniques applied to this problem. In Olshausen and Field (1996) the posterior is approximated by a delta function at its MAP value. Thus at every iteration of learning and for every data vector the maximization in (4.14) needs to be performed[6]. In Lewicki and Sejnowski (2000) it was argued that the approximation can be significantly improved if a Gaussian distribution around this MAP value was constructed by matching the

[6]In fact the situation is slightly better using a variational point of view. One can show that one can also improve a bound on the log-likelihood by jointly maximizing over $\mathbf{s}$ and A. We also note that an extra condition on the mixing matrix is needed to prevent it from collapsing to 0.

second derivatives locally (i.e. the Laplace approximation). Attias (1999) and Girolami (2001) use a variational approach which replaces the true posterior with a tractable approximation which is itself adapted to better approximate the posterior. Finally, MCMC sampling methods, such as Gibbs sampling may be employed to solve the inference problem approximately (Olshausen and Millman, 2000).

A notably different variation on the generative theme is the Bayesian approach taken by Hyvärinen and Inki (2002). There, a prior distribution $p(A)$ over possible mixing matrices A is introduced which favors orthogonal basis-vectors (columns of A). They argue that the role of the Jacobian $|\det W| = 1/|\det A|$ in (4.5) is precisely to encourage orthogonality among basis-vectors[7] and that it is therefore a reasonable assumption to remove this Jacobian in favor of the extra prior. The resultant expression is then easily extended to over-complete representations.

We want to stress that causal generative models will almost always lead to very difficult inference problems. In contrast, generating unbiased samples from the distribution $p(\mathbf{x})$ is relatively straightforward, since we first sample source values independently from their priors and subsequently sample the input variables according to the conditional Gaussian in (4.13).

4.3.2 The Information Maximization Approach

In section 4.2 an information maximization approach to ICA was discussed for the simple case when the number of inputs is equal to the number of sources and no noise is assumed on the inputs. A natural question is whether that objective can be generalized to over-complete representations. One possibility advocated by Shriki et al. (2002) is to define again the parametrized non-linear mapping (4.7) between inputs and outputs and to maximize their mutual information (which amounts to maximizing the entropy of the outputs). Note that this approach is best classified as a filtering approach, and that inputs are mapped one-to-one onto a *subset* of all possible outputs, i.e. the image of that mapping forms a lower dimensional manifold in output

[7] It is assumed that the columns of A have unit lengths.

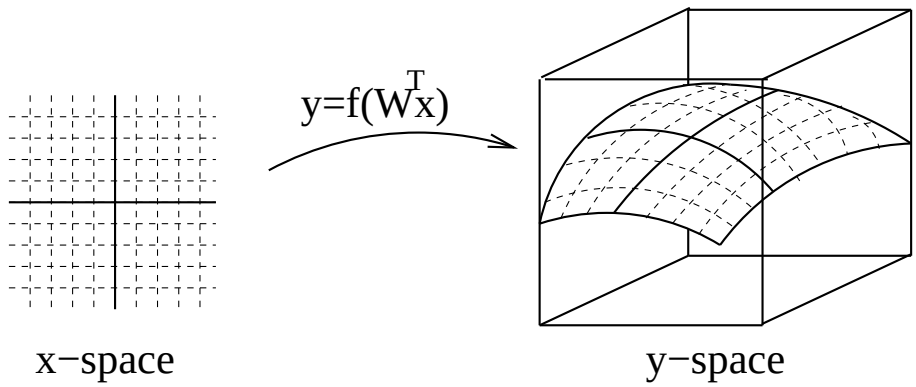


Figure 4.4: Mapping used by the information maximization approach given by (4.7).

space (see figure 4.4). Shriki et al. (2002) showed that this objective translates into maximizing the following expression for the entropy,

$$H(\mathbf{y}) = -\int d\mathbf{x}\, p^0(\mathbf{x}) \log \frac{p^0(\mathbf{x})}{\sqrt{\det(J(\mathbf{x})^T J(\mathbf{x}))}} \tag{4.15}$$

where $J(\mathbf{x})$ is the Jacobian defined in (4.10), and $p^0(\mathbf{x})$ is the data distribution.

4.4 Energy-Based Models

By interpreting ICA as a filtering model of the inputs, we now describe a very different way of generalizing ICA to over-complete representations. Energy-based models (EBM) preserve the computationally attractive property that the features $\mathbf{u}$ are simple deterministic functions of the inputs, instead of stochastic latent variables as in a causal generative model. As a consequence, even in the over-complete setting the posterior $p(\mathbf{u}|\mathbf{x})$ collapses to a point, which stands in sharp contrast to over-complete causal models which define a posterior *distribution* over the sources. In fact, for over-complete EBMs, not all feature values are allowed, since not all values lie in the image of the mapping from $\mathbf{x}$ to $\mathbf{u}$. This is similar to the information maximization approach but very different from the causal generative approach where all source values are allowed.

Let $u_i(\mathbf{x}; \mathbf{w}_i)$ be the mapping from $\mathbf{x}$ to feature u_i with parameters $\mathbf{w}_i$. The features are

used for assigning an energy $E(\mathbf{x})$, to each possible observation vector $\mathbf{x}$, as follows

$$E(\mathbf{x}) = \sum_{i=1}^{M} E_i(u_i(\mathbf{x}; \mathbf{w}_i)) \tag{4.16}$$

The probability of $\mathbf{x}$ is defined in terms of its energy through the Boltzmann distribution[8]

$$p(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z} = \frac{e^{-\sum_i E_i(u_i(\mathbf{x};\mathbf{w}_i))}}{Z} \tag{4.17}$$

where Z denotes the normalization constant (or partition function),

$$Z = \int_{\mathbf{x}'} e^{-E(\mathbf{x}')}\, d\mathbf{x}' \tag{4.18}$$

Standard ICA with non-Gaussian priors $p_i(s_i)$ is implemented by having the same number of sources as input dimensions and using

$$u_i(\mathbf{x}, \mathbf{w}_i) = \mathbf{w}_i^T \mathbf{x} \qquad\qquad E_i(u_i) = -\log p_i(u_i) \tag{4.19}$$

Furthermore, in this special case of standard ICA the normalization term in (4.17) is tractable and simplifies to

$$Z = \left| \frac{1}{\det(W)} \right| \tag{4.20}$$

where the columns of W are the filters $\mathbf{w}_i$.

The above energy-based model suggests thinking about ICA as a filtering model, where observations are linearly *filtered*, instead of as a causal generative model, where independent sources are linearly *mixed*. Hinton and Teh (2001) interpreted these filters as linear constraints, with the energies serving as costs for violating the constraints. Using energies corresponding to heavy-tailed distributions with a sharp peak at zero means that the constraints should be "frequently approximately satisfied", but will not be strongly penalized if they are grossly violated.

In this new approach it is very natural to include more constraints than input dimensions. Note however, that the *marginal independence* among the sources which was a modelling

[8]We note that the additive form of the energy leads to a product form for the probability distribution, which was called a "product of experts" (PoE) model in (Hinton, 2002).

assumption for over-complete causal models, is no longer true for the features in the EBMs in general. Instead, since the posterior $p(\mathbf{u}|\mathbf{x})$ reduces to a point, the features *given* the inputs are trivially independent,

$$p(\mathbf{u}|\mathbf{x}) = \prod_i \delta(u_i - \hat{u}_i(\mathbf{x}, \mathbf{w}_i)) \tag{4.21}$$

The semantics of such probabilistic models is consistent with that of undirected graphical models as depicted in figure 4.3b. The above means that inference in EBMs is trivial. On the other hand, sampling from the distribution $p(\mathbf{x})$ is difficult and involves MCMC in general. This is precisely opposite to causal generative models where inference is hard but sampling easy.

4.4.1 Relating EBMs to Causal Generative ICA

We will now discuss how the proposed over-complete EBMs relate to the causal generative approach to ICA. In the previous section we have already argued that when the number of input dimensions matches the number of features, an EBM is strictly equivalent to standard ICA as described in section 4.2. In the following we will now assume that there are more features than input dimensions (i.e. $M > D$).

Consider an ICA model where we have added $M - D$ auxiliary input dimensions $\mathbf{z}$. We will denote the total input space by $\mathbf{v} = [\mathbf{x}, \mathbf{z}]$. We will also add additional filters from the new $\mathbf{z}$ variables to all features and denote them by F^T, i.e. the total filter matrix is now $G^T = [W^T|F^T]$. We will assume that the new filters are chosen such that G is invertible, i.e. that the new enlarged space is fully spanned. For this enlarged ICA model we can again write the probability distribution as in (4.5), here being

$$p(\mathbf{x}, \mathbf{z}) = |\det G| \prod_{i=1}^{M} p_i(\mathbf{w}_i^T \mathbf{x} + \mathbf{f}_i^T \mathbf{z}) \tag{4.22}$$

where $\mathbf{f}_i$ are the columns of F. Next, we write the probability density for the conditional distribution,

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z})} = \frac{p(\mathbf{x}, \mathbf{z})}{\int p(\mathbf{x}', \mathbf{z})\, d\mathbf{x}'} \tag{4.23}$$

where the $|\det G|$ terms have cancelled. If we choose "observed" values for the auxiliary

variables $\mathbf{z} = 0$ then this can be written as

$$p(\mathbf{x}|\mathbf{z} = 0) = \frac{\prod_i p_i(\mathbf{w}_i^T \mathbf{x})}{\int \prod_i p_i(\mathbf{w}_i^T \mathbf{x}') \, d\mathbf{x}'} \tag{4.24}$$

The above is of course just an EBM where the partition function is given by,

$$Z = \int \prod_i p_i(\mathbf{w}_i^T \mathbf{x}') \, d\mathbf{x}' \tag{4.25}$$

Note that the above derivation is independent of the precise choice of the filters F^T as long as they span the extra dimensions.

In the previous section we have seen that an EBM may be interpreted as an undirected graphical model with conditional independence of the features given the inputs. From the above discussion we may conclude that we can also interpret the EBM as a conditional distribution $p(\mathbf{x}|\mathbf{z} = 0)$ on a directed graphical model, where $M - D$ auxiliary variables $\mathbf{z}$ have been clamped at 0. (see figure 4.3c). By clamping the extra nodes we introduce dependencies among the features through the phenomenon of "explaining away". In other words, the features are constrained by the requirement that $\mathbf{z} = 0$, which creates dependencies.

4.4.2 Relating EBMs to Information Maximization

In section 4.3.2 we saw that in the information maximization approach to over-complete representations one maximizes the entropy (4.15). The fact that the quantity $\sqrt{\det(J(\mathbf{x})^T J(\mathbf{x}))}$ in that equation is not, in general, normalized as it is in the complete case, prevents expression (4.15) from being a negative KL divergence. If we therefore define the probability density,

$$p(\mathbf{x}) = \frac{1}{Z} \sqrt{\det(J(\mathbf{x})^T J(\mathbf{x}))} \tag{4.26}$$

where Z is the normalization constant, then minimizing the KL divergence $KL(p^0||p)$ is equivalent to maximizing the log-likelihood of the model $p(\mathbf{x})$. Importantly, $p(\mathbf{x})$ is consistent with the definition of an EBM if we choose as the energy,

$$E(\mathbf{x}) = -\log\left(\sqrt{\det(J(\mathbf{x})^T J(\mathbf{x}))}\right) = -\frac{1}{2} \operatorname{Tr}\left[\log\left(J(\mathbf{x})^T J(\mathbf{x})\right)\right] \tag{4.27}$$

The energy-based density model $p(\mathbf{x})$ in (4.26) has a simple interpretation in terms of the mapping (4.7). This mapping is depicted in figure 4.4 where it is shown that the $\mathbf{x}$-coordinates define a parametrization of the manifold. It is not hard to show that the distribution $p(\mathbf{x})$ is transformed precisely to a uniform distribution $p(\mathbf{y}) = 1/Z$ on the manifold in $\mathbf{y}$-space, where the normalization constant Z may thus be interpreted as the *volume* of this manifold. Minimizing the KL divergence $KL(p^0||p)$ can therefore be interpreted as mapping the data to a manifold in a more high-dimensional embedding space, in which the data are distributed as uniformly as possible. The relation between information maximization and the above energy-based approach is summarized by the following expression,

$$H(\mathbf{y}) = -KL(p^0(\mathbf{x})||p(\mathbf{x})) + \log \text{(Manifold-Volume)} \tag{4.28}$$

The first term describes the "fit" of the model $p(\mathbf{x})$ to data, while the second term is simply the entropy of the uniform distribution $p(\mathbf{y})$ on the manifold. Relative to the energy-based approach, maximizing the mutual information will have a stronger preference to increase the volume of the manifold, since this is directly related to the entropy of $p(\mathbf{y})$. Note that in the square case the manifold is exactly the whole image space $[0, 1]^M$, hence its volume is always fixed at 1, and (4.28) reduces exactly to the KL divergence $KL(p^0(\mathbf{x})\|p(\mathbf{x}))$.

In the over-complete case, experiments will have to decide which approach is preferable and under what circumstances.

4.5 Parameter Estimation for Energy-Based Models

In section 4.4 we proposed energy-based models as probabilistic models for over-complete representations. We did not, however, discuss how to fit the free parameters of such models (e.g. filters $\mathbf{w}_i^T$) efficiently to data. In this section we will address that issue.

First, we describe the usual maximum likelihood method of training such models. For over-complete models, we show that maximum likelihood is not a practical solution, because of the non-trivial partition function. In light of this, we propose another estimation method for energy-based models called contrastive divergence (Hinton, 2002). This is a biased method,

but we will show that the bias is acceptably small compared with the gain in efficiency in training over-complete models, and the ease with which we can generalize the method to new and more intricate models.

Let $p^0(\mathbf{x})$ be the distribution of the observed data, and $p^\infty(\mathbf{x}) = p(\mathbf{x})$ be the model distribution given in (4.17) (the reason for this notation will become apparent later in the section). We would like p^∞ to approximate p^0 as well as possible. The standard measure of the difference between p^0 and p^∞ is the Kullback-Leibler (KL) divergence:

$$KL(p^0 \| p^\infty) = \int p^0(\mathbf{x}) \log \frac{p^0(\mathbf{x})}{p^\infty(\mathbf{x})} \, d\mathbf{x} \tag{4.29}$$

Because p^0 is fixed, minimizing the KL divergence is equivalent to maximizing the log likelihood of the data under the model p^∞. For energy-based models given by (4.17), the derivative of the KL divergence with respect to a weight w_{ij} is

$$\frac{\partial KL(p^0 \| p^\infty)}{\partial w_{ij}} = E_{p^0}\left[\frac{\partial E(\mathbf{x})}{\partial w_{ij}}\right] - E_{p^\infty}\left[\frac{\partial E(\mathbf{x})}{\partial w_{ij}}\right] \tag{4.30}$$

Learning can now proceed by using the derivative in (4.30) for gradient descent in the KL divergence between the data distribution and the model distribution,

$$\Delta w_{ij} \propto -\frac{\partial KL(p^0 \| p^\infty)}{\partial w_{ij}} \tag{4.31}$$

The above update rule can be understood as lowering the energy surface at locations where there are data (first term in (4.30)) and at the same time raising the energy surface at locations where there are no data but the model predicts high probability (second term in (4.30)). This will eventually result in an energy surface with low energy (high probability) in regions where there are data present and high energy (low probability) everywhere else.

The second term on the RHS of (4.30) is obtained by taking the derivative of the log partition function (4.18) with respect to w_{ij}. In the square ICA case, the log partition function is exactly given by $\log|\det W^{-1}|$, hence the second term evaluates to the ij^{th} entry of the matrix W^{-T}. However if the model is over-complete, there is no analytic form for the partition function so exact computation is generally intractable. Instead, since the second term is an expectation under the model distribution p^∞, one possibility is to use Markov chain Monte Carlo

(MCMC) techniques to approximate the average using samples from p^∞ (see Neal, 1993). This method inherits both the advantages and drawbacks associated with MCMC sampling. The estimate obtained is consistent (i.e. the bias decreases to zero as the length of the chains is increased), and it is very easily adaptable to other more complex models. The main drawback is that the method is very expensive – the Markov chain has to be run for many steps before it approaches the equilibrium distribution p^∞ and it is hard to estimate how many steps are required. Also, the variance of the MCMC estimator is usually high. To reduce the variance many independent samples are needed, incurring additional computational costs. Therefore, estimating the derivative (4.30) accurately by MCMC sampling is slow and can be unreliable due to high variance.

However, in the following we will argue that it is unnecessary to estimate the derivatives averaged over the equilibrium distribution in order to train an energy-based model from data. Instead, we will average the derivatives over a different distribution, resulting from truncating the Markov chain after a fixed number of steps. This idea, called contrastive divergence learning, was first proposed by Hinton (2002) to improve both computational efficiency and reduce the variance at the expense of introducing a bias for the estimates of the parameters with respect to the maximum likelihood solution.

There are two ideas involved in contrastive divergence learning. The first one is to start the Markov chain at the data distribution p^0 rather than to initialize the Markov chain at some vague distribution (e.g. a Gaussian with large variances). The reason usually given for using vague initial distributions is that every mode of the equilibrium distribution has a chance of being visited by some chain. This can help to overcome a problematic feature of many Markov chains – a low mixing rate; once a chain enters a mode of the distribution it is hard to escape to a different mode. However, we argue that starting at the data distribution is preferable since the training data already contains examples from the various modes that the model distribution ought to have. Towards the end of learning, when the modes of the model distribution roughly correspond to the modes in the data distribution, the number of samples in each mode approximately matches the number of data vectors in each mode. This further reduces the vari-

ance of the derivative estimates. The main danger with this technique is that certain spurious modes devoid of actual data could be accidentally created during learning, and these may go unnoticed.

The second idea of contrastive divergence is to run the Markov chain for only a few iterations rather than until equilibrium. Because the chains are started at the data distribution, even after only a few iterations, any consistent tendency to move away from the data distribution provides valuable information that can be used to adapt the parameters of the model. Intuitively, the parameters of the model should be updated so that the Markov chain does not tend to move away from the data distribution (since we want the Markov chain to equilibrate to the data distribution).

Combining the two ideas described above and defining $p^n(\mathbf{x})$ to be the distribution of the random variable at the n^{th} iteration of the Markov chain[9], the contrastive divergence learning algorithm is implemented by using the following quantity to update the filters w_{ij},

$$\Delta w_{ij} \propto -E_{p^0}\left[\frac{\partial E(\mathbf{x})}{\partial w_{ij}}\right] + E_{p^n}\left[\frac{\partial E(\mathbf{x})}{\partial w_{ij}}\right] \tag{4.32}$$

Relative to maximum likelihood learning ((4.30) and (4.31)) we have replaced the equilibrium distribution p^∞ with p^n, and the Markov chain is initialized at the data distribution p^0. Algorithm 4.1 gives pseudo-code for contrastive divergence learning in EBMs.

Notice that in order to compute the average in the second term of (4.32) we used samples produced by Markov chains initialized at the corresponding data vectors used in the first term. This, rather than uniformly sampling the initial states of the Markov chains from the data vectors, further reduces the variance.

If in addition to the filter weights w_{ij} additional parameters are present, for instance to model the shape of the energies E_i, update rules similar to (4.33) can be used to fit them to data. For standard ICA, this would correspond to learning the shape of the prior densities.

If the model distribution p^∞ is flexible enough to perfectly model the data distribution[10]

[9]Hence the notation p^0 for the initial distribution of the Markov chain and p^∞ for the limit distribution of p^n as $n \to \infty$.

[10]In the case of finite data, we replace the data distribution by the *empirical* distribution, which is a mixture

Algorithm 4.1 Contrastive Divergence Learning for Energy-Based Models

1. Run until convergence criterion is met or run time limit exceeded:
2. Compute the gradient of the total energy with respect to the parameters and average over the data cases d_k.
3. Run MCMC samplers for n steps, starting at every data vector d_k, keeping only the last sample s_k of each chain.
4. Compute the gradient of the total energy with respect to the parameters and average over the samples s_k.
5. Update the parameters using,

$$\Delta w_{ij} = -\frac{\eta}{N}\left(\sum_{\text{data } d_k} \frac{\partial E(d_k)}{\partial w_{ij}} - \sum_{\text{samples } s_k} \frac{\partial E(s_k)}{\partial w_{ij}}\right) \tag{4.33}$$

where η is the learning rate and N the number of samples in each mini-batch.

p^0, and if we use a Markov chain that mixes properly, then contrastive divergence learning has a fixed point at the maximum likelihood solution, i.e. when $p^\infty = p^0$. This is not hard to see, since at the maximum likelihood solution, the Markov chain will not change the model distribution, which implies that the derivatives in (4.32) precisely cancel. In general however, we expect contrastive divergence learning to trade-off bias with variance (see also Williams and Agakov, 2002). Apart from this, it may also happen that for certain Markov chains spurious fixed points exist in contrastive divergence learning (for some examples see MacKay, 2001).

4.5.1 Hybrid Monte Carlo Sampling

In this section we have described in detail CD learning without reference as to which Markov chains to use. Although we have argued that it is unnecessary for the Markov chain to approach equilibrium, it is still important that we choose a Markov chain that mixes fast to reduce the

of delta-functions. In this case, any smooth model distribution will not be able to perfectly fit the empirical data distribution and the above argument fails. In fact, we may expect to incur a certain bias with respect to the maximum likelihood solution.

bias in CD learning. In this subsection we will describe hybrid Monte Carlo (HMC), a very flexible class of Markov chains. HMC mixes sufficiently fast that we have used it in all our experiments on EBMs reported in this chapter. See section 5 of Neal (1993) for an in-depth description of HMC and related techniques and theory.

HMC can be thought of as a type of Metropolis-Hastings sampling. In Metropolis-Hastings sampling, a proposal distribution is used to sample a candidate x' given a previous state x, and a rejection rule is used to accept or reject x' as the next state (if it is rejected, x is used as the next state instead). To keep the acceptance rate reasonably high, the candidate typically involves a small change to the previous state; as a result, Metropolis-Hastings sampling often exhibits random walk behaviour. The idea of HMC is to move the candidate x' away from x while keeping the acceptance rate reasonably high using a deterministic dynamical system. This reduces the random walk behaviour and improves the mixing rate of the Markov chain.

Given an EBM with energy function $E(x)$, HMC requires both the ability to evaluate $E(x)$, and the ability to calculate the gradient $\frac{\partial E}{\partial x}(x)$ for each x. Both can be easily calculated for an EBM. We introduce a momentum variable $\mathbf{k}$ of equal dimensionality as $\mathbf{x}$, and define the Hamiltonian as

$$H(\mathbf{x}, \mathbf{k}) = E(\mathbf{x}) + \frac{1}{2} \|\mathbf{k}\|^2 \tag{4.34}$$

The Hamiltonian can be understood as the total energy of the system, with $\frac{1}{2}\|\mathbf{k}\|^2$ the kinetic energy and $E(\mathbf{x})$ the potential energy. The distribution over both $\mathbf{x}$ and $\mathbf{k}$ is then

$$p(\mathbf{x}, \mathbf{k}) = \frac{1}{Z} e^{-H(\mathbf{x},\mathbf{k})} = \frac{1}{Z} e^{-E(\mathbf{x})} e^{-\frac{1}{2}\|\mathbf{k}\|^2} = p^{\infty}(\mathbf{x}) N(\mathbf{k}; 0, I) \tag{4.35}$$

where $N(\mathbf{k}; 0, I)$ means $\mathbf{k}$ is Gaussian distributed with zero mean and unit covariance. Hamiltonian dynamics is given by the following dynamical equations

$$\frac{dx}{d\tau} = +\frac{\partial H}{\partial k} = k \tag{4.36}$$

$$\frac{dk}{d\tau} = -\frac{\partial H}{\partial x} = -\frac{\partial E}{\partial x} \tag{4.37}$$

where τ is a time parameter. Intuitively, the dynamics describe the movements of a ball of unit mass on a frictionless surface. The ball has position x and momentum k, and at position x the

ball is located at a height giving it potential $E(x)$. It can be shown that the distribution $p(\mathbf{x}, \mathbf{k})$ is invariant to evolving $\mathbf{x}$ and $\mathbf{k}$ according to the dynamics (4.36, 4.37).

Algorithm 4.2 Hybrid Monte Carlo Sampling

1. At iteration t, we start with the previous sample x_{t-1}. Let $x(0) = x_{t-1}$.
2. Sample initial momentum vector $k(0)$ from a zero mean unit covariance Gaussian.
3. Simulate Hamiltonian dynamics for time $L\epsilon$, using L leapfrog steps of size ϵ each. For $\tau = 0, \epsilon, 2\epsilon, \ldots, (L-1)\epsilon$:

$$\begin{aligned} k(\tau + \tfrac{\epsilon}{2}) &\leftarrow k(\tau) - \frac{\epsilon}{2}\frac{\partial E}{\partial x}\left(x(\tau)\right) \\ x(\tau + \epsilon) &\leftarrow x(\tau) + \epsilon k(\tau + \tfrac{\epsilon}{2}) \\ k(\tau + \epsilon) &\leftarrow k(\tau + \tfrac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial E}{\partial x}\left(x(\tau + \epsilon)\right) \end{aligned} \tag{4.38}$$

   The number of leapfrog steps L and the step size ϵ are user-defined parameters.
4. Accept either $x(L\epsilon)$ or $x(0)$ as the new sample x_t:

$$x_t = \begin{cases} x(L\epsilon) & \text{with probability } \min\left(1, e^{-H(x(L\epsilon),k(L\epsilon))+H(x(0),k(0))}\right) \\ x(0) & \text{otherwise} \end{cases} \tag{4.39}$$

The HMC Markov chain is used to obtain a sample (x, k) from $p(\mathbf{x}, \mathbf{k})$. Marginalizing $p(\mathbf{x}, \mathbf{k})$ to $\mathbf{x}$, we see that x will be a sample from $p^\infty(\mathbf{x})$ if (x, k) is a sample from $p^\infty(\mathbf{x}, \mathbf{k})$. Algorithm 4.2 describes an iteration of HMC. The intuition is that in iteration t of HMC, given a previous state x_{t-1} and k_{t-1}, we first discard k_{t-1} and sample a new momentum vector k'_t. This is simply Gibbs sampling (since $p(\mathbf{x}, \mathbf{k})$ factors in (4.35)) and leaves $p(\mathbf{x}, \mathbf{k})$ invariant. Then Hamiltonian dynamics is applied for some fixed amount of time $L\epsilon$, where at time 0 we start with $x(0) = x_{t-1}$ and $k(0) = k'_t$. The resulting $x(L\epsilon)$ and $k(L\epsilon)$ is taken as the next state of the Markov chain, i.e. $x_t = x(L\epsilon)$, $k_t = k(L\epsilon)$. Since this leaves $p(\mathbf{x}, \mathbf{k})$ invariant as well, the whole operation leaves $p(\mathbf{x}, \mathbf{k})$ invariant. Of course we actually simulate the Hamiltonian dynamics by using the leapfrog discretization (4.38). The leapfrog discretization is designed

to preserve phase space volume, i.e. each measurable set (x, k) of values for position x and momentum k is mapped to another set *with the same measure*. This is crucial to preserve the invariance of $p(\mathbf{x}, \mathbf{k})$ under the discretized dynamics. However the leapfrog discretization still introduces errors which is then corrected by the final rejection step. Notice that the third step of the leapfrog step can be combined with the first step of the next leapfrog step:

$$k(\tau + \tfrac{\epsilon}{2}) \leftarrow k(\tau) - \frac{\epsilon}{2}\frac{\partial E}{\partial x}(x(\tau)) = k(\tau - \tfrac{\epsilon}{2}) - \epsilon\frac{\partial E}{\partial x}(x(\tau)) \tag{4.40}$$

The error in the discretization is affected by the step size (it scales as $O(\epsilon^2)$) which in turn affects the acceptance rate. We find that the optimal step sizes used are quite sensitive to the particular problem and to the phase of learning (e.g. at the start of learning the model distribution p^∞ is quite vague and we can use a large step size, while at the end of learning a small step size is needed). Hence instead of using the step size as a parameter, we use the acceptance rate as a parameter, and adjust the step size so that the actual acceptance rate is in the vicinity of the desired rate. The adjustment is done after each CD step (which might involve multiple HMC steps) to avoid affecting the equilibrium property of the Markov chain. We use a simple adjustment rule:

$$\epsilon \leftarrow \epsilon - \eta\,((\text{desired rate}) - (\text{empirical rate})) \tag{4.41}$$

where η is another user-defined parameter.

4.6 Experiment: Blind Source Separation

In collaboration with Max Welling, we assessed the performance of contrastive divergence as a learning algorithm. We compared a HMC implementation of contrastive divergence with an exact sampling algorithm as well as the Bell and Sejnowski (1995) algorithm on a standard "blind source separation" problem. The model has the same number of input and source

dimensions[11], and the energy of the model is defined as

$$E_i(s_i) = -\log(\sigma(s_i)(1-\sigma(s_i))) \tag{4.42}$$

This model is strictly equivalent to the noiseless ICA model with sigmoidal outputs used by Bell and Sejnowski (1995).

The data consisted of sixteen five-second stereo CD recordings of music, sampled at 44.1 kHz[12]. Each recording was monoized, down-sampled by a factor of 5, randomly permuted over the time-index and rescaled to unit variance. The resulting 88436 samples in 16 channels were linearly mixed using the standard *instamix* routine with $b = 0.5$ (1 on the diagonal and $1/9$ off the diagonal)[13], and whitened before presentation to the various learning algorithms. The whitening process is a simple linear transformation (called the ZCA) which makes the covariance matrix identity and has been observed to improve ICAs speed of convergence.

The HMC implementation of contrastive divergence uses 1 outer loop step of HMC simulation to sample from $p^1(\mathbf{x})$, which in turn consists of 30 leapfrog steps, with the step sizes adapted at the end of each simulation so that the acceptance rate is about 90%. This is algorithm HMC in the following. For noiseless ICA, it is possible to sample efficiently from the true equilibrium distribution using the causal generative view. This is used in EQUIL. To be fair, we used a number of samples equal to the number of data vectors in each mini-batch. We can also compute the partition function using (4.20), and evaluate the second term of (4.30) exactly. This is precisely Bell and Sejnowski's algorithm and was implemented in EXACT.

Parameter updates were performed on mini-batches of 100 data vectors. The learning rate was annealed from 0.05 down to 0.0005 in 10000 iterations of learning[14], while a momentum factor of 0.9 was used to speed up convergence. The initial weights were sampled from a Gaussian with a standard deviation of 0.1.

During learning we monitored the Amari-Distance[15] to the true unmixing matrix. In figures

[11]Note however that recovering more sound sources than input dimensions (sensors) is not possible with our energy-based model, since the feautures are not marginally independent.

[12]Prepared by Barak Pearlmutter and available at `http://sweat.cs.unm.edu/~bap/demos.html`.

[13]Available at `http://sound.media.mit.edu/ica-bench`.

[14]2000 iterations each at 0.05, 0.025, 0.005, 0.0025 and 0.0005.

[15]The Amari-Distance (Amari et al., 1996) measures a distance between two matrices A and B up to permuta-

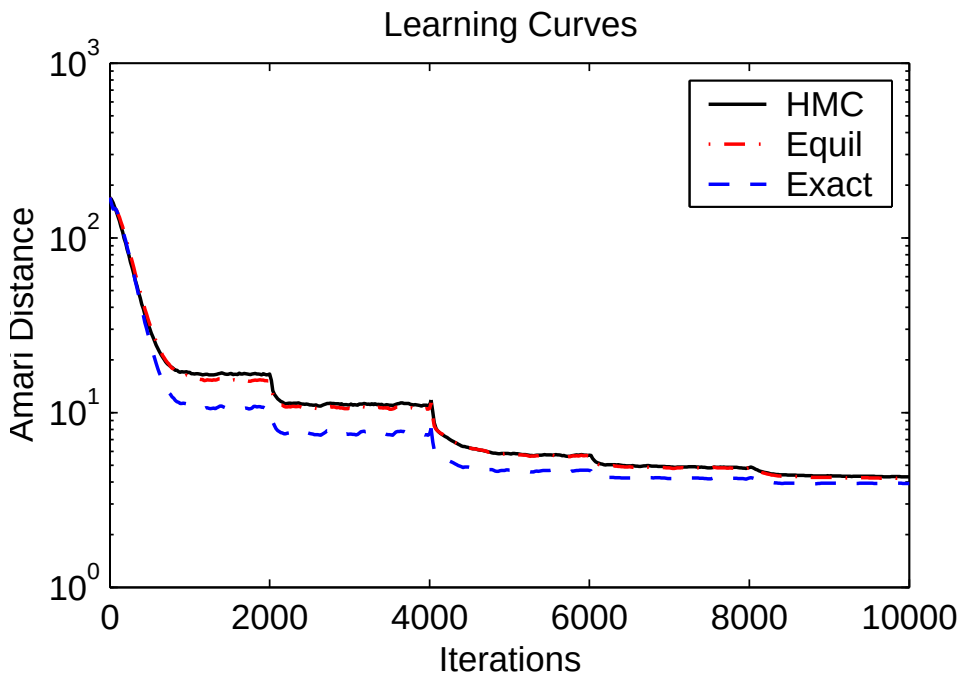


Figure 4.5: Evolution of the Amari Distance for the various algorithms on the blind source separation problem, averaged over 100 runs. Note that HMC converged just as fast as the exact sampling algorithm EQUIL, while the exact algorithm EXACT is only slightly faster. The sudden changes in Amari distance are due to the annealing schedule.

4.5 and 4.6 we show the results of the various algorithms on the sound separation task. The figures show that the deterministic method EXACT performs slightly better than the sampling methods HMC and EQUIL, probably due to the variance induced by the sampling. More importantly, it shows that learning with brief sampling (HMC) performs about as well as learning with samples from the equilibrium distribution (EQUIL). The main conclusion of this experiment is that we do not need to sample from the equilibrium distribution in order to learn the filters $\mathbf{W}$. This validates the ideas behind contrastive divergence learning.

tions and scalings: $\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{|(AB^{-1})_{ij}|}{\max_k|(AB^{-1})_{ik}|}+\frac{|(AB^{-1})_{ij}|}{\max_k|(AB^{-1})_{kj}|}\right)-2N^2$.

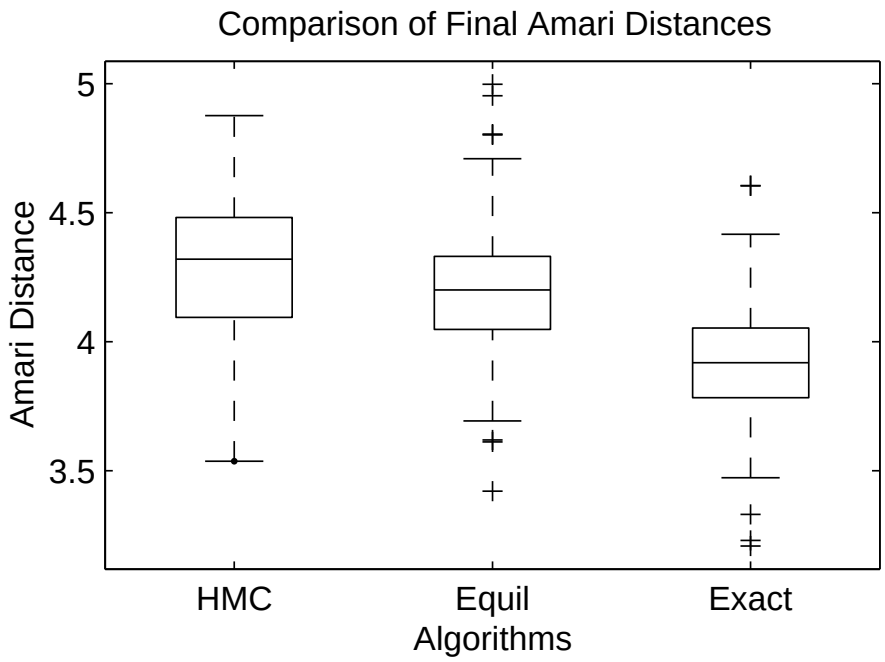


Figure 4.6: Final Amari-Distances for the various algorithms on the blind source separation problem, averaged over 100 runs. The boxes have lines at the lower quartile, median, and upper quartile values. The whiskers show the extent of the rest of the data. Outliers are denoted by "+".

4.7 Experiments: Feature Extraction

We present examples of the features delivered by our algorithm on several standard datasets. Firstly we demonstrate performance on typical ICA tasks of determining an over-complete set of features of speech and natural images. Then, we show the algorithm applied to the CEDAR cdrom dataset of hand-written digits and lastly, we present the feature vectors learned when the algorithm is applied to the FERET database of human faces.

For all the experiments described in this section we use an energy function of the form:

$$E_i(u_i(\mathbf{x}, \mathbf{w}_i)) = \gamma_i \log\left(1 + (\mathbf{w}_i^T \mathbf{x})^2\right) \tag{4.43}$$

which corresponds to modelling the data with a product of one-dimensional student-t distributions with $(2\gamma - 1)$ degrees of freedom (Hinton and Teh, 2001). This energy function was

chosen for its simplicity yet versatility in describing super-Gaussian distributions. However, the algorithmic formulation allows the use of arbitrary energy functions and results may be improved by a more systematic tailoring of the energy function to particular datasets.

4.7.1 Speech

Max Welling performed an experiment to test whether the model could extract meaningful filters from speech data. He used recordings of 10 male speakers from the TIMIT database, uttering the sentence:

"Don't ask me to carry an oily rag like that."

The sentences were down-sampled to 8kHz and 50000, 12.5ms segments (each segment corresponding to 100 samples) were extracted from random locations. Before presentation to the learning algorithm the data was centred (i.e. the mean was removed) and whitened. The features were trained using contrastive divergence with one step of HMC sampling consisting of 20 leapfrog steps. Mini-batches of size 100 were used, while the learning rate was annealed from 0.05 to 0.0005 over 20000 iterations. The filters were initialized at small random values and momentum was used to speed up parameter learning.

In figure 4.7 we show 10 of the 200 features in the whitened domain together with their power spectra. Recall that since there are 2 times more filters extracted as dimensions in the input space, the energy-based model is no longer equivalent to a causal ICA model. Figure 4.8 shows the distribution of power over time and frequency. There seems to be interesting structure around 1.5kHz, where the filters are less localized and more finely tuned in frequency than average. This phenomenon is also reported by Abdallah and Plumbley (2001).

4.7.2 Natural Image Patches

We tested our algorithm on the standard ICA task of determining the "independent" components of natural images. The data set used is the `imlog`[16] data set of van Hateren and van der

[16] Available at `ftp://hlab.phys.rug.nl/pub/samples/imlog`.

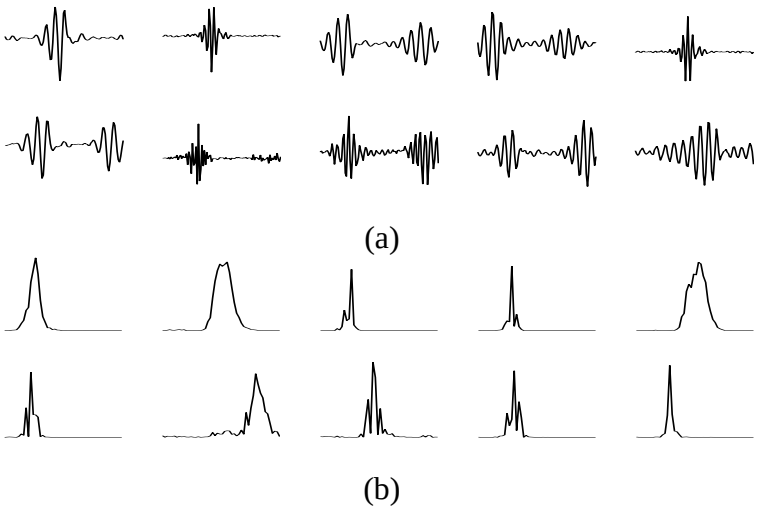


Figure 4.7: (a) Filters found by the $2\times$ over-complete EBM. The 5 filters in the first row are the ones with largest power, indicating that they represent important features. The 5 filters in the second row are randomly drawn from the remaining 195 filters. (b) Corresponding power-spectra.

Schaaf (1998). The logarithm of the pixel intensities was first taken and then the image patches were centred and whitened. There were 122880 patches and each patch was 16×16 in size. We trained a network with $256 \times 3 = 768$ features, using contrastive divergence with 1 step of HMC sampling consisting of 30 leapfrog steps. The step size was adaptive so that the acceptance rate is approximately 90%. Both $\mathbf{w}_i$ and γ_i are unconstrained, but a small weight decay of 10^{-4} was used for $\mathbf{w}_i$ to encourage the features to localize. The $\mathbf{w}_i$'s were initialized to random vectors of length 1, while the γ_i's were initialized at 1. Both $\mathbf{w}_i$ and γ_i were trained with a learning rate of 0.01 and momentum factor of 0.9. We found however that the result is not sensitive to the settings of these parameters. A random sample of 144 learned features in the whitened domain is shown in figure 4.9. They were roughly ordered by increasing spatial frequency. By hand, we counted a total of 19 features which have not localized either in the spatial or frequency domain. Most of the other features can be described well with Gabor functions. To further analyze the set of learned filters, we fitted a Gabor function of the form used by Lewicki and Olshausen (1999) to each feature and extracted parameters like frequency,

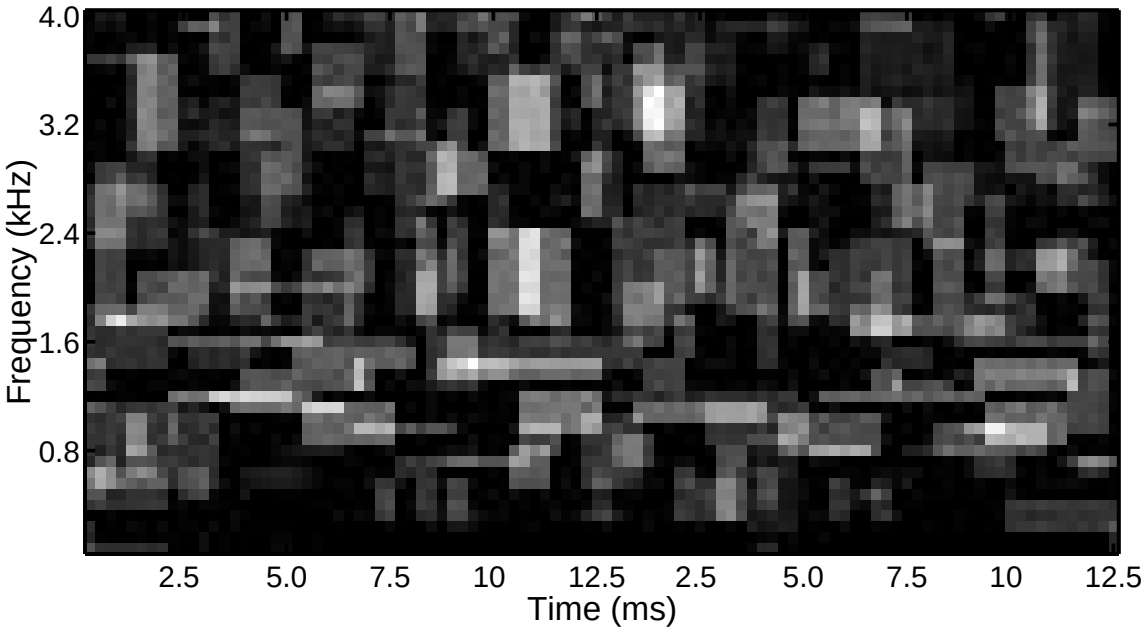


Figure 4.8: Distribution of power over time and frequency for the learned speech features. First the envelope of each filter (the absolute value of its Hilbert transform) was computed and squared. Next, the squared envelope and the power spectrum were thresholded by mapping all values greater than half the peak value to one and the rest to zero. Gaps smaller than 6 samples in time and 3 samples in frequency were filled in. Finally, the outer product of the two "templates" were computed, weighted by the total power of the filter, and added to the diagram.

location and extent in the spatial and frequency domains. These are summarized in figures 4.10 and 4.11, and show that the filters form a nice tiling of both the spatial and frequency domains. We see from figures 4.9 and 4.11 that filters are learned at multiple scales, with larger features typically being of lower frequency. However we also see an over emphasis of horizontal and vertical filters. This effect has been observed in previous papers (van Hateren and van der Schaaf, 1998, Lewicki and Olshausen, 1999), and is probably due to pixellation.

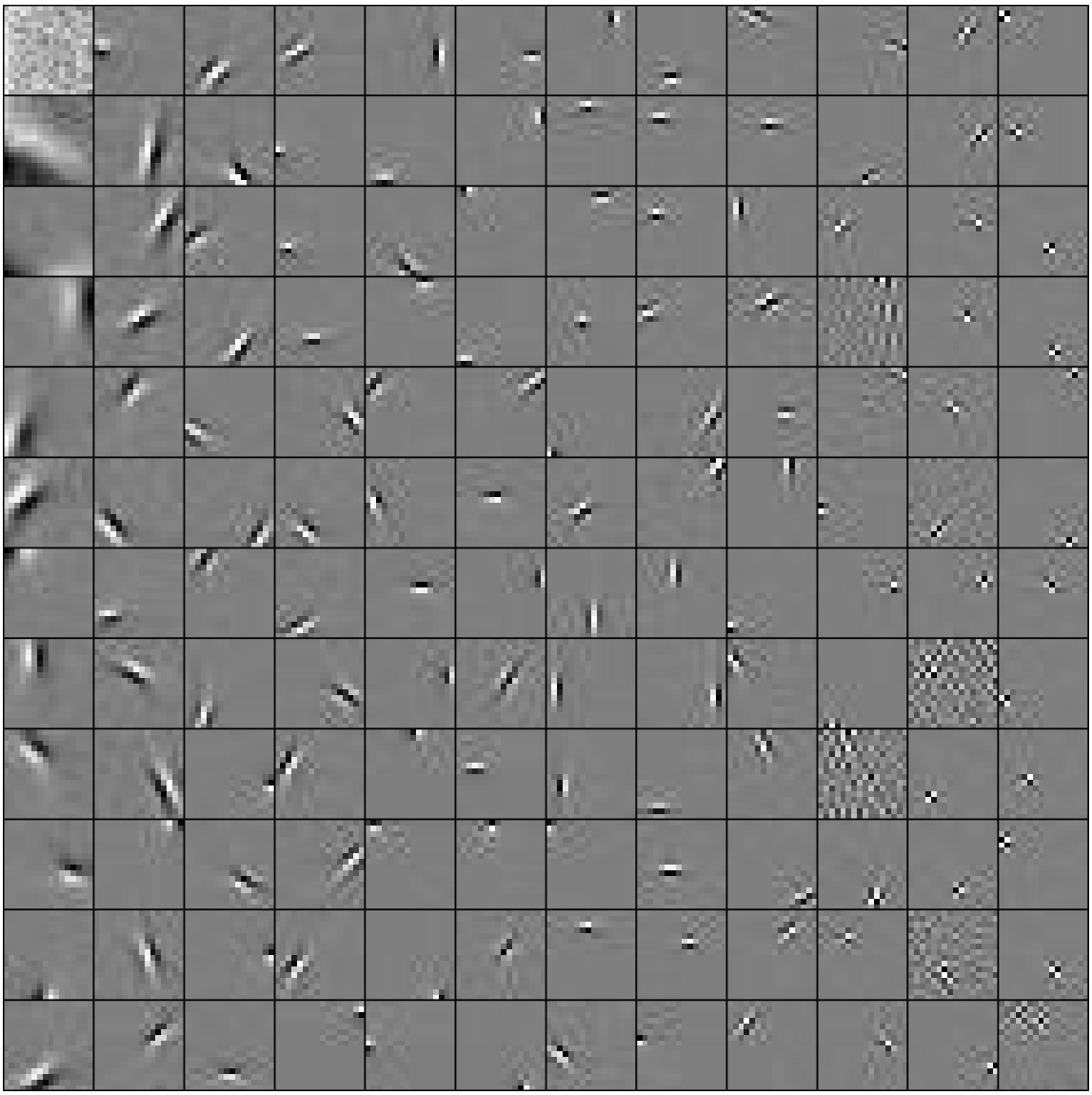


Figure 4.9: Learned filters for natural images.

4.7.3 CEDAR Digits

In collaboration with Simon Osindero, we applied EBMs to images of hand-written digits. We used 16x16 real valued digits from the "br" set on the CEDAR cdrom #1. There are 11000 digits available, divided equally into 10 classes. The mean image from the entire dataset was subtracted from each datum, and the digits were whitened. A network with 361 features was trained in the same manner as for natural image patches.

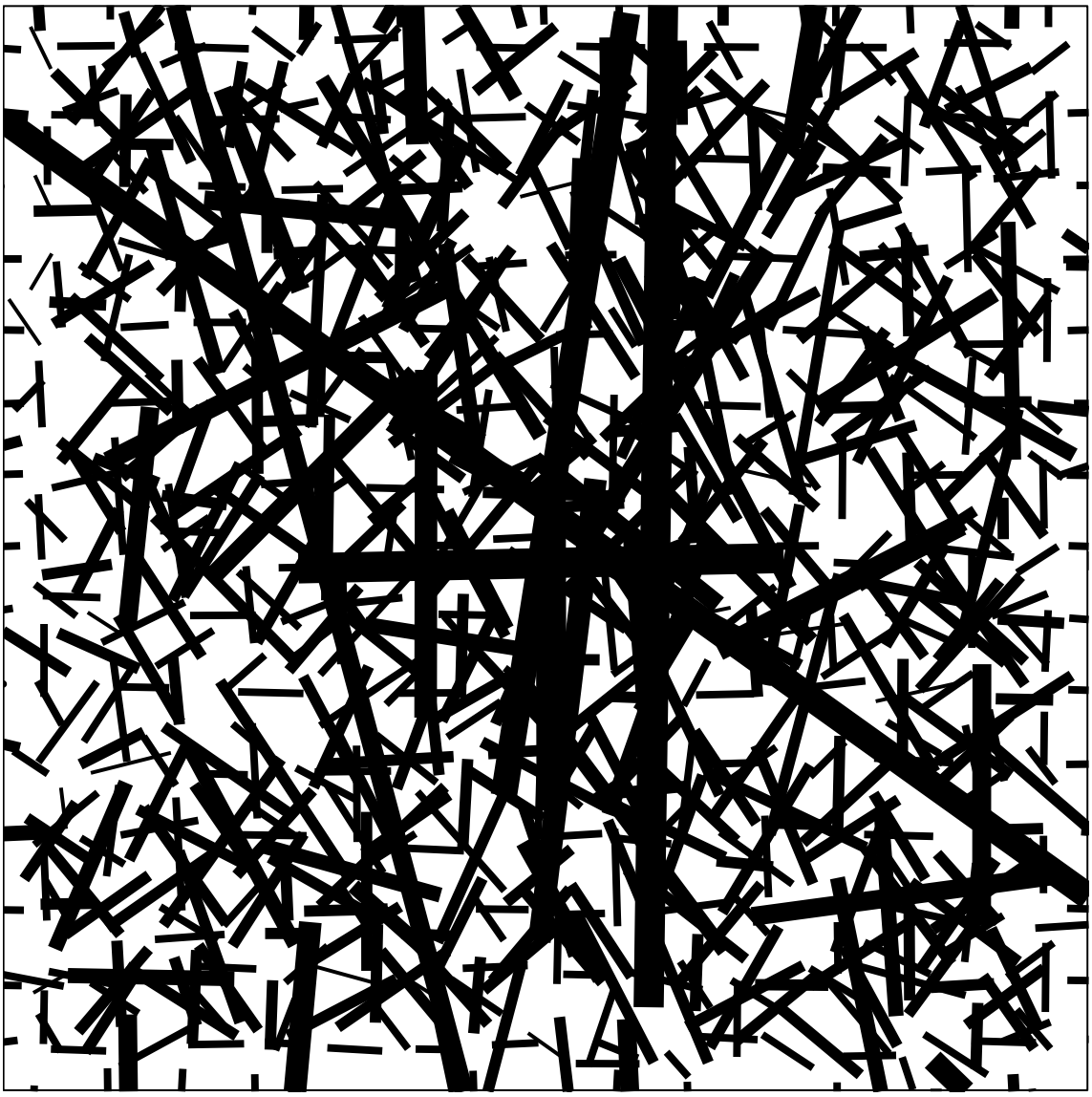


Figure 4.10: The spatial layout and size of the filters learned on natural image patches, which are described by the position and size of the bars.

A random subset of learned filters is shown in figure 4.12. To make it easier to discern the structure of the learned filters, we present them in the whitened domain rather than in pixel space. We note the superficial similarity between these filters and those found from the natural scene experiments. However, in addition to straight edge filters we also see several curved filters. We interpret the results as a set of 'stroke' detectors, modelling a space of strokes that

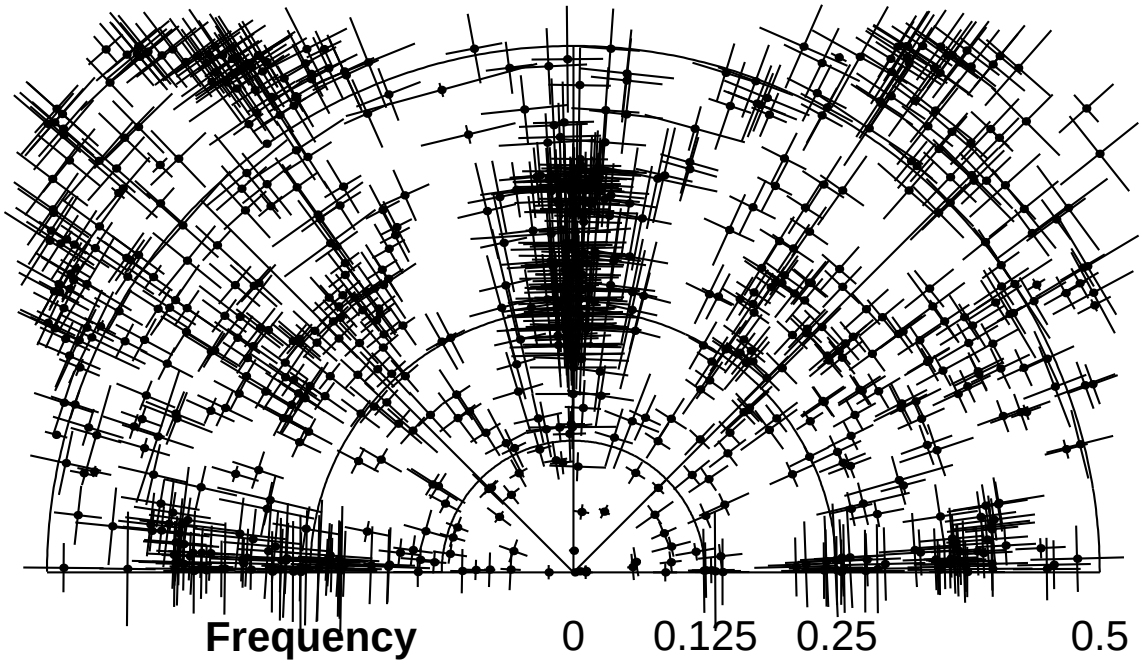


Figure 4.11: A polar plot of frequency tuning and orientation selectivity of the filters learned on natural image patches, with the centre of each cross at the peak frequency and orientation response, and crosshairs describing the $1/16$-bandwidth.

gives rise to the full digit set.

4.7.4 FERET Faces

In collaboration with Simon Osindero, we applied EBMs to the database of frontal face images used in the 1996 FERET face recognition evaluation (Phillips et al., 1997). The data was first pre-processed in the standard manner by aligning the faces, normalizing the pixel intensities and cropping a central oval shaped region[17]. Then as an additional pre-processing step we centred the data and retained the projections onto the leading 256 principal components as the input dimensions to the algorithm. The projections are also normalized so that they have variance one. We trained an EBM with 361 features using contrastive divergence with HMC

[17]The code for pre-processing is part of a project on evaluating face recognition algorithms by Ross Beveridge and Bruce Draper. It is available at
`http://www.cs.coloradostate.edu/evalfacerec/index.htm`.

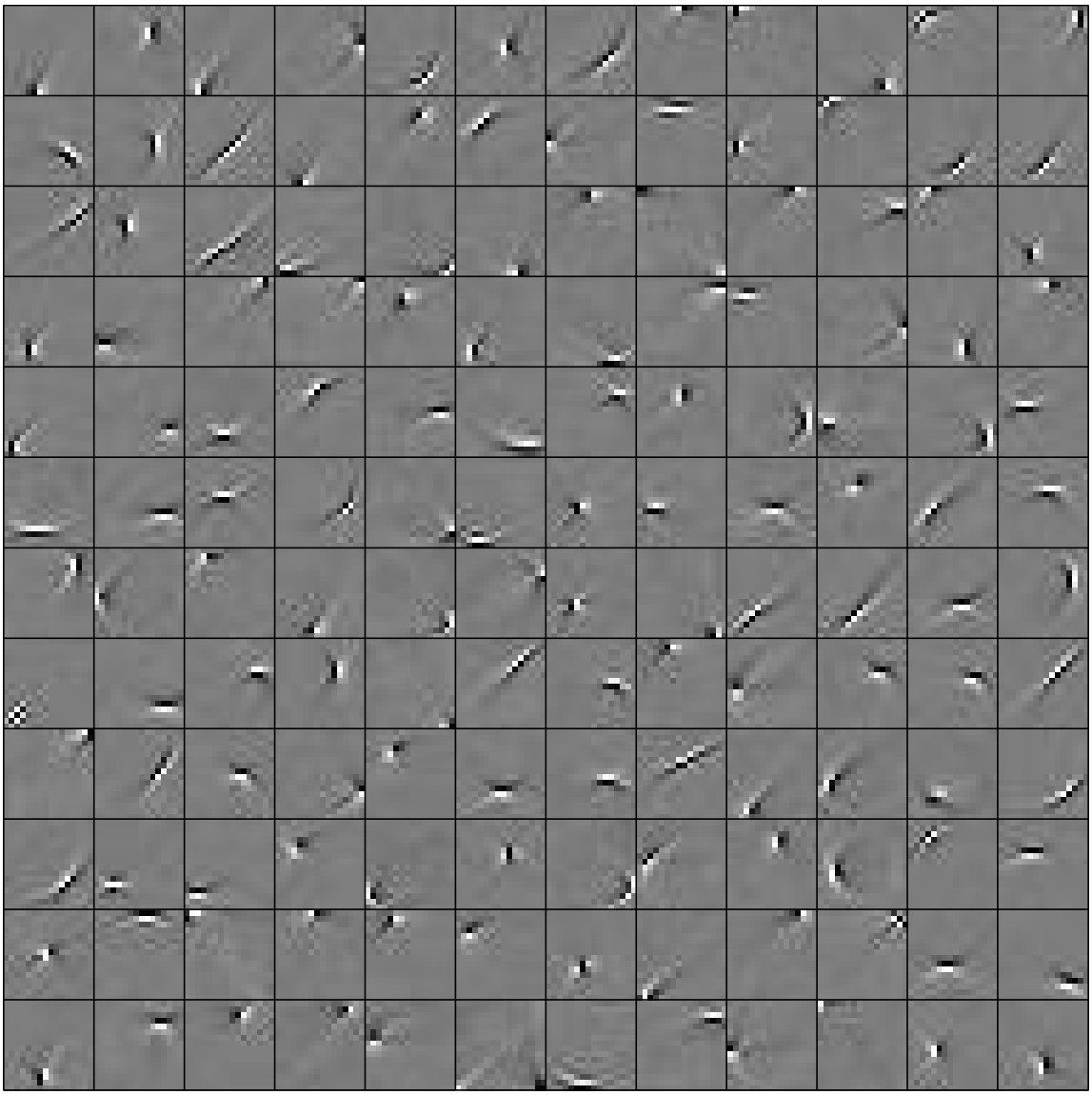
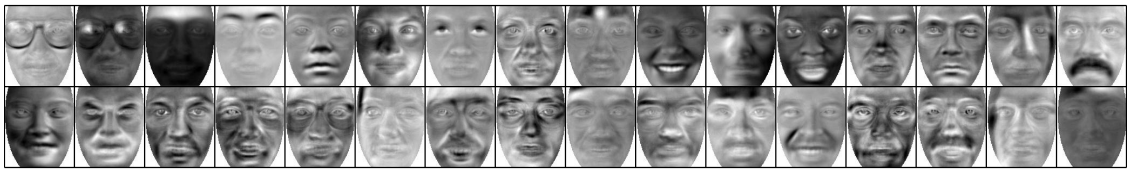


Figure 4.12: Learned filters for CEDAR digits. Filters are plotted in whitened space for clarity.

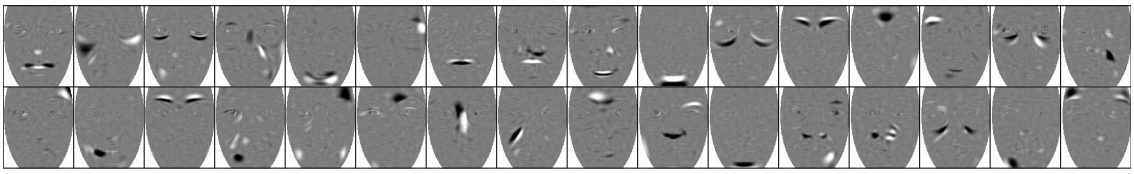
sampling (3 sets of 20 leapfrog steps). Both the $\mathbf{w}_i$ and γ_i were unconstrained. The $\mathbf{w}_i$ were initialized with values from a Gaussian distribution and the vector norm of each weight vector was rescaled to 1. The γ_i were initialized at 1. A learning rate of 0.005 was used for the $\mathbf{w}_i$, whilst a learning rate of 0.001 was used for the γ_i.

Figure 4.13a shows the 32 leading eigenvectors plotted as face images, and figure 4.13b shows a subset of 32 filters as learned by our model. Many of the learned filters are somewhat

(a)



(b)



(c)

Figure 4.13: (a) 32 eigenfaces with largest eigenvalue plotted rowwise in descending eigenvalue order. (b) Subset of 32 feature vectors learned using the EBM on ordinary face data. The top row are hand picked, the bottom row randomly selected. (c) Subset of 32 feature vectors learned using architecture I, all randomly selected.

global in that most pixels have a non-zero weight. However, in addition to these global features there are also more localized features focussing on glasses, eyes, smiling mouths, moustaches etc. Furthermore, as well as global features that can perhaps be described as 'archetypical faces' (which are similar to those learned by RBMrate in figure 3.5) we also see global features which appear to mainly capture structure in the illumination of a face.

The features learned here are not as sparse as those for natural images or hand-written digits. The reasons are twofold. First since the face images are normalized the pixel values are highly correlated to each other. Second, the model imposes (through the energy terms) a sparsity constraint on the outputs of the filters, while there is no actual pressure for the filter weights themselves to be sparse. To impose sparsity on the weights we can again force them to

be non-negative. However Bartlett et al. (2002) proposed a more interesting solution (referred to as architecture I) which we use here. The idea is that we invert the roles of the filter outputs and weights, so that the same EBM will now impose the sparsity constraint on the weights instead.

We describe the technique in the context of causal generative square ICA (which as we have shown is equivalent to an EBM with equal numbers of filters and inputs). Let the matrix X have columns which correspond to training images. Square ICA can be understood as a decomposition

$$X = AU \tag{4.44}$$

where the columns of A are the basis vectors, and the columns of U are the sources which generated the images (columns in X). A sparse prior is then imposed on the entries of U. The equivalent of this for EBMs gives the features in figure 4.13b.

Taking the transpose of (4.44), we have a decomposition

$$X^T = U^T A^T \tag{4.45}$$

Now if we place a sparse prior on the entries of A, the decomposition can still be interpreted as ICA, but with columns of U^T as the basis vectors and A^T the sources instead.

When performing ICA on the columns of a matrix M (in the above context M could be X or X^T), it is important that the number of columns be greater than the number of rows (note that at least one of X or X^T will not satisfy this condition). If the number of rows is greater, then the columns only span a subspace of the whole space. In this case we should perform ICA only within the spanned subspace by projecting the columns down onto the subspace. A related issue is that learning in ICA is more efficient if we first whiten the columns of M so that they have unit covariance. The typical way to handle both issues is by using PCA (Bartlett et al., 2002). Let n be the desired number of principal components. We take n to be less than the number of columns and rows. Let the columns of P be the first n principal components of the columns of M, and the diagonal matrix Λ have the associated eigenvalues along its diagonal. We construct the projected matrix $M' = \Lambda^{-\frac{1}{2}} P^T M$. The columns of M' are whitened and

represents the original columns of M in the projected space. ICA can now be applied to the matrix M'.

The above framework is in terms of causal generative square ICA. Making use of the relationship between square ICA and EBMs, we can extend this framework easily to the over-complete case using EBMs. In particular, we apply over-complete EBMs to the columns of $M' = \Lambda^{-\frac{1}{2}} P^T M$, where $M = X^T$. Figure 4.13c illustrates the results when we take $n = 256$. The features shown are actually the outputs of the filters learned by EBM (rows of A^T in (4.45)). This approach leads to features that are all highly localized in space, since the model imposes a sparse prior on the outputs of the filters. Our results are again qualitatively similar to those described in Bartlett et al. (2002).

4.8 Discussion

In this chapter we have re-interpreted the standard ICA algorithm as an energy-based model and studied its extension to over-complete representations. We have shown that parameters of an EBM, such as the filter weights and those parametrizing the energy function, can be efficiently estimated using contrastive divergence learning. Through a number of experiments on standard data sets we have shown that EBMs can efficiently extract useful features in high dimensions. In conclusion we believe that EBMs provides a flexible modelling tool which can be trained efficiently to uncover useful structure in continuous-valued data.

Contrary to causal generative models for over-complete ICA, the features of an EBM exhibit marginal dependencies. The advantage of allowing these dependencies in the model is fast inference. In causal generative models, the assumption of marginal independence often leads to intractible inference which needs to be approximated using some iterative, data dependent scheme. The role of these iterations can be understood as suppressing the "activity" of less relevant features, thus producing a sparse code. Therefore, for causal generative models, over-complete representations are expected to produce very compact (or sparse) codes, a fact which is often emphasized as desirable (Olshausen and Field, 1997). Perhaps surprisingly,

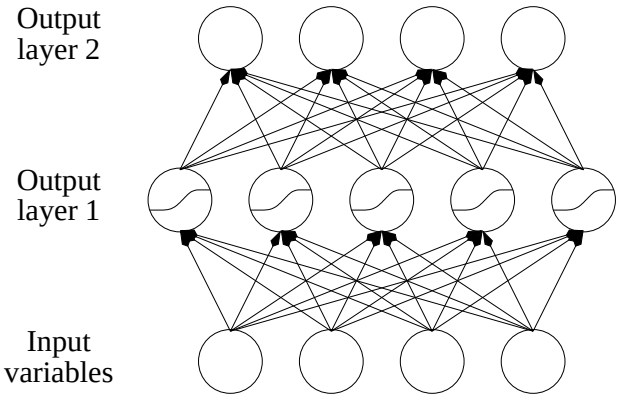


Figure 4.14: Architecture of a hierarchical non-linear energy-based model. Non-linearities are indicated by sigmoidal units in output layer 1. Energies can be contributed by output variables in both layers, and the number of output variables need not correspond to the number of input variables.

we have shown that such a slow iterative process is in fact not needed to produce sparse and over-complete representations.

However the above does suggest enriching EBMs with inhibitory lateral connections to achieve the goal of further suppressing less relevant features in order to produce an even sparser representation. Preliminary experiments using a mean field approach to implement these lateral inhibitions have been successful in learning good density models, but are slow due to the iterative optimization for every data case.

Another powerful generalization of EBMs is a hierarchical non-linear architecture in which the output activities are computed with a feed-forward neural network (figure 4.14) and each layer may contribute to the total energy. In a recent paper by Welling et al. (2003), a two layer model was studied where the second layer performed a local averaging of the non-linearly transformed activities of the first layer. This resulted in a topographic ordering of the filters, where orientation, location and frequency are changing smoothly from one filter to the next. This work builds upon the topographic ICA work of Hyvärinen and Hoyer (2001) and there is an interesting relationship between the two layer EBM and the approximation used to train the topographic ICA model in Hyvärinen and Hoyer (2001).

To fit multi-layer EBMs to data, backpropagation is used to compute gradients of the energy with respect to both the data vector (to be used in HMC sampling), and the weights (to be used for weight updates). Since this algorithm applies backpropagation in an unsupervised setting and combines it with contrastive divergence learning we have named it "contrastive backpropagation" (Hinton et al., 2002). Indeed, the contrastive backpropagation learning procedure is quite flexible. It puts no constraints other than smoothness on the activation functions or the energy functions[18]. The procedure can be easily modified to use recurrent neural networks that contain directed cycles by running each forward pass for some predetermined number of steps and defining the energy to be any smooth function of the time history of the activations. Backpropagation through time (Rumelhart et al., 1986, Werbos, 1990) can then be used to obtain the required derivatives. The data-vector can also change during the forward pass through a recurrent network. This makes it possible to model sequential data, such as video sequences, by running the network forward in time for a whole sequence and then running it backward in time to compute the derivatives required for HMC sampling and for updating the weights.

The energy-based approach to ICA we have presented stems from previous work on PoEs. In fact an EBM is a type of PoE that defines a density over continuous-valued domains. Because no discretization is involved EBMs are more natural models to use than the RBMrate model of chapter 3. In future work we would like to explore the various extensions to EBMs mentioned above, and apply EBMs to face recognition to compare it against RBMrate and other standard face recognition algorithms.

Acknowledgements and Remarks

An earlier version of this chapter was written in collaboration with Max Welling, Simon Osindero and Geoffrey Hinton and has been submitted as a paper to the Journal of Machine Learning Research Special Issue on Independent Component Analysis (Teh et al., 2003). Some of the

[18]There is also the necessary assumption that the energy is defined such that the Boltzmann distribution is normalizable.

reported experiments have been performed solely or partially by my collaborators, and I would like to thank them for kindly letting me report them here. In particular, Max Welling performed experiments on feature extraction from speech, as well as the first version of the blind source separation experiment which appeared in Hinton et al. (2001). While both Simon Osindero and I have run feature extraction experiments on digits and faces and obtained the same qualitative results, the figures reported here are from his experiments.

We thank Yair Weiss and Peter Dayan for first pointing out a possible relationship between EBMs and ICA. We thank the Ontario Government and the Gatsby Charitable Foundation for funding.

Chapter 5

The Bethe Free Energy for Inference

Along with variational techniques and MCMC sampling, loopy belief propagation has emerged in recent years as a powerful alternative for approximate inference. It was found that the fixed points of loopy belief propagation correspond exactly to stationary points of an approximate free energy called the Bethe free energy (Yedidia et al., 2001). Unfortunately loopy belief propagation does not always converge. This chapter takes the obvious next step and proposes an alternative to loopy belief propagation that will always converge to a stationary point of the Bethe free energy. We call this algorithm unified propagation and scaling because it makes use of an interesting relationship between loopy belief propagation, and the classical iterative scaling algorithm. We show in a number of toy experiments the applicability of unified propagation and scaling.

5.1 Introduction

Belief propagation (BP) is a standard algorithm to perform exact inference on tree structured graphical models (Pearl, 1988, Lauritzen and Spiegelhalter, 1988). It is an iterative algorithm whereby messages encoding evidence are passed among neighbouring nodes. When applied to graphs with cycles, this "loopy BP" algorithm is not guaranteed to converge, nor are the computed marginal distributions necessarily exact even if it converges (Pearl, 1988). This can

be understood as "double counting" evidence when messages are passed from a node back to itself via a cycle (Weiss, 1997). However, it was found empirically that loopy BP does often converge, and the produced estimates of the marginal distributions are often surprisingly accurate (Murphy et al., 1999).

Loopy BP was first discovered in the coding community[1] when the decoding algorithms of turbo codes and Gallager's low density parity check codes were found to be loopy BP on particular graphical models (Frey and MacKay, 1997, MacKay and Neal, 1996). This advance made possible other codes that approach Shannon's information theoretic limit (McEliece et al., 1998), and has motivated interest in analyzing and understanding the behaviour and accuracy of the loopy BP algorithm, both in the context of error correction decoding and elsewhere.

Initial analysis of loopy BP in terms of "unwrapping" networks revealed some interesting results (Weiss, 1997, 2000). For example, that in networks with a single loop, loopy BP will always converge and the maximum posterior assignment will always be correct even though the estimated marginals will be overly confident. Recent work by Tatikonda and Jordan (2002) along this line, where loopy BP is related to Gibbs measures on infinite trees, has also yielded strong result on the conditions under which loopy BP will converge.

Another recent analysis technique interprets loopy BP as tree-based reparameterization of the distribution represented by the graphical model (Wainwright et al., 2002b). This line of analysis has led to useful bounds on the errors obtained by loopy BP if it converges.

Yet another theoretical foundation of loopy BP was the discovery that fixed points of the algorithm correspond exactly to stationary points of the Bethe free energy (Yedidia et al., 2001). This Bethe free energy can be understood as an approximation to the exact Gibbs free energy (Georges and Yedidia, 1991, Welling and Teh, 2001). In this respect there are strong relations between loopy BP and variational and mean field methods (Jordan et al., 1998, Saad and Opper, 2001). This line of research has led to generalized BP algorithms whose fixed points are stationary points of the more sophisticated Kikuchi free energy (Yedidia et al., 2001).

Empirically, it has been found that the fixed points of loopy BP are local minima of the

[1] In the coding literature loopy BP is often called the sum-product algorithm or the generalized distributive law.

Bethe free energy, leading to conjectures that loopy BP should always converge to local minima of the Bethe free energy (Yedidia et al., 2001). Recently, this conjecture has been proven by Heskes (2003). He also showed that not every local minima need be a stable fixed point of loopy BP.

Since loopy BP converges to local minima of the Bethe free energy, and the Bethe free energy is a reasonable approximation to the true Gibbs free energy, the obvious next step is to derive an algorithm which actually minimizes the Bethe free energy and always converges. This is the aim of this chapter. To do so, we have to solve a generalization of inference based on minimizing an approximate KL divergence. The algorithm we derive is closely related to both loopy BP and iterative scaling (IS). This *unified propagation and scaling* (UPS) algorithm will then reduce to a convergent alternative to loopy BP when generalized inference reduces to ordinary inference.

In section 5.2 we describe the Markov networks we will be using throughout the chapter and in section 5.3 we describe inference and review some of the background on loopy BP and the Bethe free energy. In section 5.4 we introduce our generalization of inference and the iterative scaling (IS) algorithm. In section 5.5, we propose the Bethe free energy as an approximation to the KL divergence and derive fixed point equations to perform approximate generalized inference. We also show in what sense our fixed point equations are related to loopy BP and IS. In section 5.6 we describe various algorithms to minimize the Bethe free energy, including our unified propagation and scaling (UPS) algorithm. Section 5.7 shows experimental simulations on the various algorithms, and section 5.8 concludes.

5.2 Markov Networks

Consider a discrete pairwise Markov network G with node set N and edge set E. For each $i \in N$ let N_i denote the neighbours of i. Let $(ij) \in E$ denote edges of the graph. Let X_i be the variable associated with node $i \in N$. Denote the domain of values of X_i by $\mathbb{X}_i$ and let $x_i \in \mathbb{X}_i$ be a value of X_i. For a node set $n \subset N$ let $X_n = \{X_i\}_{i \in n}$ and $X = X_N$; similarly for

$x_n, x, \mathbb{X}_n, \mathbb{X}$ etc.

Let the single and pairwise potentials be $\psi_i(x_i)$, $\psi_{ij}(x_i, x_j)$. Then the distribution represented by G is

$$P(X = x) = \frac{1}{Z_P} \prod_{(ij)\in E} \psi_{ij}(x_i, x_j) \prod_{i\in N} \psi_i(x_i) \tag{5.1}$$

where Z_P is the normalization constant (partition function). We will use a slightly different parameterization of $P(X)$. Let $\phi_{ij}(x_i, x_j) = \psi_{ij}(x_i, x_j)\psi_i(x_i)\psi_j(x_j)$, $\phi_i(x_i) = \psi_i(x_i)$, and n_i be the number of neighbours of node i. Then

$$P(X = x) = \frac{1}{Z_P} \prod_{(ij)\in E} \phi_{ij}(x_i, x_j) \prod_{i\in N} \phi_i(x_i)^{1-n_i} \tag{5.2}$$

In the following we will drop the cumbersome notation $P(X = x)$ and instead use $P(x)$.

In the rest of this chapter we will be dealing with inference on pairwise Markov networks. This can be generalized to other graphical models by using the reductions of Weiss and Freeman (2001).

5.3 Ordinary Inference

Let $V \subset N$ be the set of "visible" nodes and $H = N\backslash V$ be the set of "hidden" nodes. X_V will be the random variables that we observe and we would like to infer or approximate the posterior distribution over X_H given observations on X_V. For ordinary inference, where we observe that the visible nodes X_V take on one particular value x_V, the effect of the observation can be absorbed into the potentials as

$$\psi_i(x_i)^{new} \leftarrow \psi_i(x_i) \prod_{k\in N_i\cap V} \psi_{ik}(x_i, x_k) \tag{5.3}$$

Then the posterior distribution over X_H is again given by (5.1) where the visible nodes V and neighbouring edges have been removed from the graph and the potentials have been altered using (5.3). For this reason, when we discuss ordinary inference, we will not explicitly refer to visible and hidden nodes in the graph. Rather, we will take ordinary inference as determining the marginal distributions of interest in the distribution P given in (5.2). In section 5.4 we

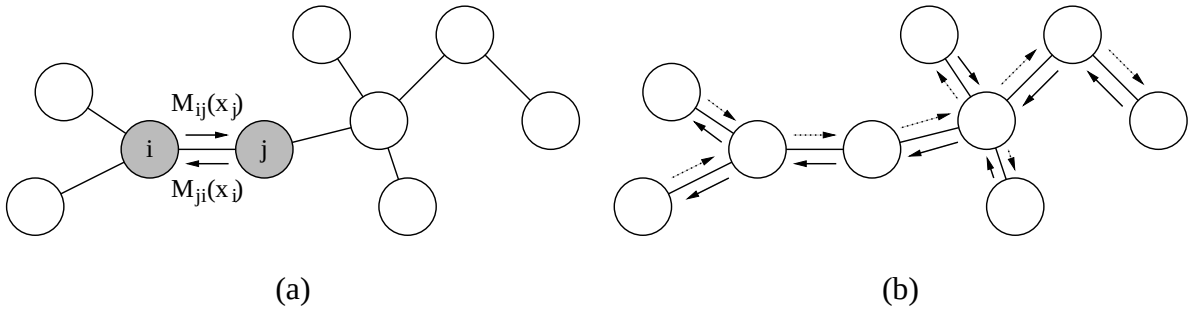


Figure 5.1: Belief propagation on tree structured Markov networks. (a) $M_{ij}(x_j)$ is the message sent from node i to node j, while $M_{ji}(x_i)$ is the message from j to i. (b) The messages represented by the solid and dotted arrows do not affect one another.

will generalize the notion of inference using minimum divergence ideas. In that case we will explicitly refer to visible and hidden nodes.

5.3.1 Belief Propagation

Belief propagation (BP) is an inference procedure for tree structured Markov networks. It is an iterative algorithm whereby messages are passed between neighbouring nodes. See figure 5.1a. The message $M_{ij}(x_j)$ from node i to node j encodes the contribution of the subtree rooted at i (nodes to the left of j in the figure) as it affects the distribution of X_j. Messages are updated with the update rule

$$M_{ij}(x_j) \leftarrow \sum_{x_i} \frac{\phi_{ij}(x_i, x_j)}{\phi_j(x_j)} \prod_{k \in N_i \setminus j} M_{ki}(x_i) \tag{5.4}$$

where the messages $M_{ki}(x_i)$ encode the contribution of the rest of the subtree to the distribution of x_i. After the messages have converged, the marginal distributions are computed as

$$P(x_i) = \phi_i(x_i) \prod_{k \in N_i} M_{ki}(x_i) \tag{5.5}$$

$$P(x_i, x_j) = \phi_{ij}(x_i, x_j) \prod_{k \in N_i \setminus j} M_{ki}(x_i) \prod_{l \in N_j \setminus i} M_{lj}(x_j) \tag{5.6}$$

A simple but important property of the message updates (5.4) is that the messages going in either direction on the tree are independent of each other. This is illustrated in figure 5.1b, where

the solid and dashed arrows represent the messages going in either direction. A consequence of this property is that any scheduling of the message updates (5.4) will always converge on a tree. By waiting until all incoming messages $M_{ki}(x_i)$ have been updated before updating $M_{ij}(x_j)$, we obtain a particularly efficient scheduling, in which each message needs to be updated only once before the algorithm converges. One particular instantiation of this schedule are the well-known CollectEvidence and DistributeEvidence routines.

5.3.2 Bethe Free Energy

When belief propagation is applied on graphs with cycles (loopy BP), Yedidia et al. (2001) showed that its fixed points correspond exactly to stationary points of the Bethe free energy. The Bethe free energy is an approximation to the Gibbs free energy which arose in statistical physics (Georges and Yedidia, 1991, Welling and Teh, 2001).

Let the beliefs $b_{ij}(x_i, x_j)$ and $b_i(x_i)$ be estimates of the pair-wise and single site marginal distributions of the desired distribution P. The beliefs need to satisfy the following marginalization and normalization (MN) constraints:

$$\sum_{x_j} b_{ij}(x_i, x_j) = b_i(x_i) \qquad \forall\, i \in N, j \in N_i \qquad \sum_{x_i} b_i(x_i) = 1 \qquad \forall\, i \in N \tag{5.7}$$

Let $Q = \{b_{ij}(x_i, x_j)\}_{(ij)\in E} \cup \{b_i(x_i)\}_{i\in N}$. The Bethe free energy is defined as

$$\mathcal{F}_{bethe}(Q) = \sum_{(ij)\in E} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{\phi_{ij}(x_i, x_j)} + \sum_{i\in N} (1 - n_i) \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\phi_i(x_i)} \tag{5.8}$$

To impose the MN constraints we use Lagrange multipliers $\lambda_{ji}(x_i)$ and γ_i. The Lagrangian is given by

$$\mathcal{L} = \mathcal{F}_{bethe}(Q) - \sum_{i\in N} \sum_{j\in N(i)} \sum_{x_i} \lambda_{ji}(x_i) \left(\textstyle\sum_{x_j} b_{ij}(x_i, x_j) - b_i(x_i) \right) - \sum_{i\in N} \gamma_i \left(\textstyle\sum_{x_i} b_i(x_i) - 1 \right) \tag{5.9}$$

where $N(i)$ are the neighbours of i. Setting the derivatives of the Lagrangian $\mathcal{L}$ with respect to the beliefs and Lagrange multipliers to zero, and identifying $\lambda_{ji}(x_i) \propto \sum_{k\in N_i \setminus j} \log M_{ki}(x_i)$, we derive equations (5.5, 5.6) for the beliefs (replacing $P(x_i)$ with $b_i(x_i)$ and $P(x_i, x_j)$ with $b_{ij}(x_i, x_j)$), and the fixed point updates (5.4) for the messages.

5.4 Generalized Inference

In this section we will generalize the notions of posterior distributions and inference using minimum KL divergence ideas, and show how methods for approximate inference can be adapted to this scenario.

For $i \in V$ let $\widehat{o}_i(x_i)$ be a fixed distribution over X_i. Given these "observed marginal distributions" on V, define the generalized posterior as the distribution $Q(x)$ which minimizes the KL divergence

$$KL(Q\|P) = \sum_x Q(x) \left(\log Q(x) - \log P(x)\right) \tag{5.10}$$

subject to the constraints that $Q(x_i) = \widehat{o}_i(x_i)$ for each $i \in V$. We call these constraints observational (Obs) constraints. Generalized inference is the process by which we determine the generalized posterior. To avoid confusion, we will explicitly use "ordinary inference" for normal inference, but when there is no confusion "inference" by itself will mean generalized inference in this chapter; similarly for posteriors.

Let $\delta(a-b) = 1$ if $a = b$ and 0 otherwise. Also let $x_S = \{x_i\}_{i\in S}$ for a subset of nodes S. Similarly if S is a subgraph of G.

Observation 5.4.1 *If $\widehat{o}_i(x_i) = \delta(x_i - \widehat{x}_i)$ for each $i \in V$ then the generalized posterior is $Q(x) = P\left(x_H \,|\, \widehat{x}_V\right) \prod_{i\in V} \delta(x_i - \widehat{x}_i)$.*

Proof: As $Q(x)$ has to satisfy the constraints, we have $Q(x_V) = \prod_{i\in V} \delta(x_i - \widehat{x}_i)$ hence $Q(x)$ has to have the form $Q(x) = Q(x_H|\widehat{x}_V) \prod_{i\in V} \delta(x_i - \widehat{x}_i)$. Now

$$\begin{aligned} KL(Q\|P) &= \sum_{x_H}\sum_{x_V} Q(x_H, x_V) \left(\log Q(x_H, x_V) - \log P(x_H, x_V)\right) \\ &= \sum_{x_H}\sum_{x_V} Q(x_H, \widehat{x}_V) \left(\log Q(x_H|\widehat{x}_V) + \sum_{i\in V} \log \delta(x_i - \widehat{x}_i) - \log P(x_H, x_V)\right) \\ &= \sum_{x_H} Q(x_H|\widehat{x}_V) \left(\log Q(x_H|\widehat{x}_V) - \log P(x_H|\widehat{x}_V)\right) - \log P(\widehat{x}_V) \end{aligned} \tag{5.11}$$

and now minimizing (5.11) gives $Q(x_H|\widehat{x}_V) = P(x_H|\widehat{x}_V)$. ■

The above observation shows that if the constrained marginals are delta functions, i.e. the observations are hard, then the generalized posterior reduces to a trivial extension of the ordinary posterior, hence explaining our use of the term generalized inference.

Note that these observed distributions on the nodes in V are not the same as soft evidence. In soft evidence, an observational process $P(o_i|x_i)$ is associated with each $i \in V$ and we observe o_i instead of x_i. The observation only act as a bias affecting x_i and the marginal distribution of x_i is not fixed.

5.4.1 Iterative Scaling

Since generalized inference is a constrained minimum divergence problem, a standard way of solving it is using Lagrange multipliers. For each $i \in V$ and x_i, let $\lambda_i(x_i)$ be the Lagrange multiplier enforcing $Q(x_i) = \widehat{o}_i(x_i)$. Let γ be another Lagrange multiplier enforcing the fact that $Q(x)$ should sum to one. The Lagrangian is then given by

$$\mathcal{L} = KL(Q\|P) - \sum_{i \in V} \sum_{x_i} \lambda_i(x_i) \left(Q(x_i) - \widehat{o}_i(x_i)\right) - \gamma \left(\textstyle\sum_x Q(x) - 1\right) \tag{5.12}$$

Solving for $Q(x)$ by setting the derivatives of $Q(x)$ and γ to zero, we see that the generalized posterior is given by

$$Q(x) = P(x) e^{\sum_{i \in V} \lambda_i(x_i) + \gamma - 1} \tag{5.13}$$

$$\gamma = 1 - \log \sum_x P(x) e^{\sum_{i \in V} \lambda_i(x_i)} \tag{5.14}$$

The optimal γ simply introduces a partition function to normalize $Q(x)$. Plugging this back into the Lagrangian $\mathcal{L}$, we get the dual cost function

$$\mathcal{L}' = \sum_{i \in V} \sum_{x_i} \lambda_i(x_i) \widehat{o}_i(x_i) - \log \sum_x P(x) e^{\sum_i \lambda_i(x_i)} \tag{5.15}$$

Since $KL(Q\|P)$ is convex in $Q(x)$, the dual cost $\mathcal{L}'$ is concave, its maximum coincides with the minimum of $KL(Q\|P)$ subject to the Obs constraints, and the generalized posterior is given in terms of the optimal $\lambda_i(x_i)$. Maximizing the dual cost by coordinate-wise ascent gives

the classical iterative scaling (IS) algorithm (Deming and Stephan, 1940). At each iteration of IS, a Lagrange multiplier $\lambda_i(x_i)$ is updated using the scaling update

$$e^{\lambda_i(x_i)} \leftarrow e^{\lambda_i(x_i)} \frac{\widehat{o}_i(x_i)}{Q(x_i)} \qquad \text{for each } x_i \tag{5.16}$$

Intuitively, (5.16) updates the current posterior so that the marginal $Q(x_i)$ for node i matches the given constraint $\widehat{o}_i(x_i)$. However this will undo the marginal constraints at the other nodes hence the scaling update (5.16) will have to be applied iteratively to all the nodes in V until convergence.

IS is a specific case of the generalized iterative scaling (GIS) algorithm (Darroch and Ratcliff, 1972), which updates the Lagrange multipliers for a subset $U \subset V$ of nodes using

$$e^{\lambda_i(x_i)} \leftarrow e^{\lambda_i(x_i)} \left(\frac{\widehat{o}_i(x_i)}{Q(x_i)} \right)^{1/|U|} \qquad \text{for each } x_i \text{ and } i \in U \tag{5.17}$$

We can show that GIS maximizes a lower bound on the dual cost $\mathcal{L}'$ at each step, and it converges to the maximum of $\mathcal{L}'$ (see Berger, 1997). However perhaps a more intuitive understand of GIS is that the parallel GIS steps are performing IS updates in parallel, but damping the steps such that the algorithm is still guaranteed to converge.

5.5 Approximate Generalized Inference

Notice from (5.13, 5.14) that the generalized posterior has the same form as $P(x)$ except that the potentials on V have been altered by the Lagrange multipliers. Ordinary inference is needed to compute the current marginals $Q(x_i)$ required by the scaling update (5.16). If G is singly connected, then belief propagation (BP) can be used to compute the required marginals. Otherwise, exact inference or sampling algorithms like Monte Carlo Markov chain (MCMC) can be used, but usually are computationally taxing. Alternative approximate inference algorithms like variational methods and loopy BP can be used instead to estimate the required marginals. Although being much more efficient, they produce biased estimates, potentially leading to the overall IS not converging. For example, consider a two node Boltzmann machine, with weight 5 and biases -2.5, and the desired means on both nodes are 0.3. Then if either naive mean field

or naive TAP equations are used to estimate the marginals required, IS will not converge. Further, even if IS did converge, we do not have much theoretical understanding of the accuracy of the overall algorithm.

A more principled approach is to first approximate the KL divergence, then derive algorithms to minimize the approximation subject to the Obs constraints. For example, using variational approximations (Jordan et al., 1998), we upper bound the KL divergence by variational means and by assuming that $Q(x)$ comes from a tractable parametric family, and then minimizing the upper bound with respect to the variational parameters with Obs constraints. The upper bound is chosen so that this minimization is tractable. Note that this approach is generally not the same as using variational approximations within an IS outer loop.

In the next section, we describe a Bethe free energy approximation to the KL divergence. Fixed point equations for minimizing the Bethe approximation can then be derived. The fixed point equations reduce to BP updates at hidden nodes, and to IS updates at observed nodes. As a consequence, using loopy BP to approximate the marginals required by the IS outer loop turns out to be a particular scheduling of the fixed point equations. Because the Bethe free energy is fairly well understood, and is quite accurate in many regimes (Murphy et al., 1999, Yedidia et al., 2001, Welling and Teh, 2001), we conclude that IS with loopy BP is a viable approximate generalized inference technique. However, in section 5.6 we describe better algorithms for approximate generalized inference based upon the Bethe free energy.

5.5.1 Bethe Approximation

In the Bethe approximation, we first assume that $Q(x)$ can be factored as follows:

$$Q(x) \approx \prod_{(ij)} b_{ij}(x_i, x_j) \prod_i b_i(x_i)^{1-n_i} \tag{5.18}$$

where the beliefs have to satisfy the MN constraints. Note that $Q(x)$ need not be normalized and need not have the beliefs as its marginals. However it is exact if $P(x)$ and hence $Q(x)$ are tree structured. Plugging this approximation into the KL divergence and regrouping terms, we

get

$$KL(Q\|P) \approx \mathcal{F}_{bethe}(Q) + \log Z_P \tag{5.19}$$

where Z_P is the partition function of P. The Bethe approximation is an approximation to the KL divergence which only accounts for pair-wise correlations between neighbouring variables. However unlike variational approximations, the Bethe approximation is not an upper bound in general.

Since $\log Z_P$ is fixed, we wish to minimize $\mathcal{F}_{bethe}(Q)$ subject to the MN and Obs constraints. Again we use Lagrange multipliers $\lambda_{ji}(x_i)$ to impose the marginalization constraints. We can also use Lagrange multipliers to impose the normalization and observational constraints, but this reduces to simply keeping $b_{ij}(x_i, x_j)$ and $b_i(x_i)$ normalized, and keeping $b_i(x_i) = \widehat{o}_i(x_i)$ fixed for $i \in V$. We shall ignore these for clarity. The resulting Lagrangian is

$$\mathcal{L} = \mathcal{F}_{bethe}(Q) - \sum_i \sum_{j \in N(i)} \sum_{x_i} \lambda_{ji}(x_i) \left(\sum_{x_j} b_{ij}(x_i, x_j) - b_i(x_i) \right) \tag{5.20}$$

Setting derivatives of $\mathcal{L}$ with respect to $b_{ij}(x_i, x_j)$, $b_i(x_i)$ and $\lambda_{ji}(x_i)$ to zero, we get

Theorem 5.5.1 *Subject to the MN and Obs constraints, every stationary point of $\mathcal{F}_{bethe}$ is given by*

$$b_{ij}(x_i, x_j) \propto \phi_{ij}(x_i, x_j) e^{\lambda_{ji}(x_i) + \lambda_{ij}(x_j)} \tag{5.21}$$

$$b_i(x_i) \propto \phi_i(x_i) e^{\frac{1}{n_i - 1} \sum_{j \in N(i)} \lambda_{ji}(x_i)} \tag{5.22}$$

where the Lagrange multipliers are fixed points of the following updates:

$$e^{\lambda_{ji}(x_i)} \leftarrow \prod_{k \in N(i) \setminus j} \sum_{x_k} \frac{\phi_{ik}(x_i, x_k)}{\phi_i(x_i)} e^{\lambda_{ik}(x_k)} \qquad \text{for } i \notin V,\ j \in N(i) \tag{5.23}$$

$$e^{\lambda_{ji}(x_i)} \leftarrow \frac{\widehat{o}_i(x_i)}{\sum_{x_j} \phi_{ij}(x_i, x_j) e^{\lambda_{ij}(x_j)}} \qquad \text{for } i \in V,\ j \in N(i) \tag{5.24}$$

5.5.2 Relationship to IS and BP

The fixed point updates (5.23, 5.24) are closely related to both BP and IS. Identifying messages as $M_{ij}(x_j) = \sum_{x_i} \frac{\phi_{ij}(x_i,x_j)}{\phi_j(x_j)} e^{\lambda_{ji}(x_i)}$, we get the BP updates

$$M_{ij}(x_j) \leftarrow \sum_{x_i} \frac{\phi_{ij}(x_i, x_j)}{\phi_j(x_j)} \prod_{k \in N(i) \backslash j} M_{ki}(x_i) \tag{5.25}$$

This was first shown by Yedidia et al. (2001) with a different identification of $e^{\lambda_{ji}(x_i)} = \prod_{k \in N(i)\backslash j} M_{ki}(x_i)$. From (5.23) we see that this identification is equivalent to ours.

Rewriting (5.24) in terms of messages as well we find,

$$M_{ij}(x_j) \leftarrow \sum_{x_i} \psi_{ij}(x_i, x_j) \frac{\widehat{o}_i(x_i)}{M_{ji}(x_i)} \qquad \text{for } i \in V, j \in N(i) \tag{5.26}$$

We can extend the physical analogy and understand (5.26) as a message "bouncing" step, in which messages going into an observed node get bounced back and are altered in the process. If $\widehat{o}_i(x_i) = \delta(x_i - \widehat{x}_i)$ is a hard observation, then (5.26) reduces to $M_{ij}(x_j) \leftarrow \psi_{ij}(\widehat{x}_i, x_j)$ so that instead of bouncing back, messages going into node i get absorbed.

Alternatively, we can understand the message bouncing steps as IS updates.

Theorem 5.5.2 *Let $i \in V$. Updating each $\lambda_{ji}(x_i)$ for $j \in N(i)$ using (5.24) is equivalent to updating $\lambda_i(x_i)$ using (5.16), where we identify*

$$Q(x_i) \propto \phi_i(x_i) \prod_{j \in N(i)} M_{ji}(x_i) \qquad e^{\lambda_i(x_i)} = \left(\frac{\phi_i(x_i)}{\widehat{o}_i(x_i)} \right)^{n_i - 1} \prod_{j \in N(i)} e^{\lambda_{ji}(x_i)} \tag{5.27}$$

Theorem 5.5.2 states the unexpected result that scaling updates (5.16) are just fixed point equations to minimize $\mathcal{F}_{bethe}$. Further, the required marginals $Q(x_i)$ are computed using (5.23), which is exactly loopy BP. Hence using loopy BP to approximate the marginals required by IS is just a particular scheduling of the fixed point equations (5.23, 5.24).

5.6 Algorithms to Minimize the Bethe Free Energy

The fixed point equations (5.23, 5.24) determine the conditions under which the beliefs are at a stationary point of the Bethe free energy $\mathcal{F}_{bethe}$. By themselves they do not determine an

algorithm to find a stationary point of the Bethe free energy, let alone to find a local minimum. In this section we describe various algorithms which try to find either a stationary point or local minimum of $\mathcal{F}_{bethe}$.

In section 5.6.1 we describe two algorithms which directly use the fixed point updates (5.23, 5.24). These algorithms are not guaranteed to converge, nor are they guaranteed to converge to local minima even if they converge. In sections 5.6.2 and 5.6.3 we describe UPS, an algorithm which is guaranteed to converge to a local minimum or saddle point of the Bethe free energy.

5.6.1 Direct fixed point algorithms

The simplest algorithm, one inspired by Yedidia et al. (2001), is to run the fixed point updates (5.23, 5.24) and hope that they converge to a minimum of $\mathcal{F}_{bethe}$. The fixed point updates can be run in series or parallel. This is algorithm 5.1 (loopy IS). Theorem 5.5.1 only states that if loopy IS converges it will converge to a stationary point of $\mathcal{F}_{bethe}$. In simulations we find that it always gets to a good local minimum, if not the global minimum. However loopy IS does not necessarily converge, especially when the variables are strongly coupled. There are two reasons why it can fail to converge. Firstly, the loopy BP component (5.23) may fail to converge. However this is not serious as past result indicate that loopy BP often fails only when the Bethe approximation is not accurate (Welling and Teh, 2001). Secondly, the IS component (5.24) may fail to converge, since it is not run sequentially and the estimated marginals are inaccurate. We will show in section 5.7 that this is a serious problem for loopy IS.

One way to promote convergence is to damp the loopy IS updates. This works well in practice and damping is often critical to the success of loopy IS as well as of loopy BP (Murphy et al., 1999). There are two common ways to damp propagation updates – linear and geometric.

Algorithm 5.1 Loopy IS – Loopy Iterative Scaling

1. Run until convergence criterion is met or run time limit exceeded:
2. Perform the fixed point updates (5.23, 5.24).
3. If run time limit exceeded return failed to converge.

Algorithm 5.2 IS+BP – Iterative Scaling with Loopy Belief Propagation.

1. Run until convergence criterion is met or run time limit exceeded:
2. Run until convergence criterion is met or run time limit exceeded:
3. Perform the BP fixed point updates (5.23).
4. If run time limit exceeded return failed to converge.
5. Perform one scaling update (5.16).
6. If run time limit exceeded return failed to converge.

In linear damping the messages are updated as

$$M_{ij}(x_j) \leftarrow (1-\alpha)M_{ij}(x_j) + \alpha \sum_{x_i} \frac{\phi_{ij}(x_i, x_j)}{\phi_j(x_j)} \prod_{k \in N(i) \setminus j} M_{ki}(x_i) \tag{5.28}$$

where α is the damping factor, while in geometric damping we have

$$M_{ij}(x_j) \leftarrow M_{ij}(x_j)^{1-\alpha} \left(\sum_{x_i} \frac{\phi_{ij}(x_i, x_j)}{\phi_j(x_j)} \prod_{k \in N(i) \setminus j} M_{ki}(x_i) \right)^{\alpha} \tag{5.29}$$

Damping preserves the fixed points of the algorithm while making it more likely to converge. Damping can be used for scaling updates as well – in fact, GIS updates can be understood as geometric damping.

Another way to promote convergence is to mitigate the second problem by running the scaling updates (5.16) in series, and approximate the required marginals using an inner phase of loopy BP. This IS+BP algorithm is illustrated in algorithm 5.2. Theorem 5.5.2 shows that IS+BP is just a particular scheduling of loopy IS. IS+BP gets rid of convergence problems related to running scaling updates in parallel and before the marginals have converged, hence it inherits the accuracy of loopy IS while converging more often. However because we have to run loopy BP until convergence for each scaling update, IS+BP is not particularly efficient.

In the next two subsections, we describe yet another possibility – an efficient algorithm based on the same fixed point equations (5.23, 5.24) which is guaranteed to converge without damping.

5.6.2 Constraining the leaves of trees

In this section we derive UPS-T, an efficient algorithm for minimizing the Bethe free energy on a tree when the observed nodes are leaves of the tree. In the next section we will use UPS-T as a subroutine of an algorithm which minimizes the Bethe free energy for general graphs.

Suppose that G is a tree, and all observed nodes $i \in V$ are leaves of G. Since G is a tree, the Bethe free energy is exact, i.e. if the MN constraints are satisfied then $\mathcal{F}_{bethe} = KL(Q\|P)$ where $Q(x) = \prod_{(ij)} b_{ij}(x_i, x_j) \prod_i b_i(x_i)^{1-n_i}$. In particular, $\mathcal{F}_{bethe}$ is convex in the subspace defined by the MN constraints. Therefore the fixed point equations (5.23, 5.24), if run sequentially, will converge to the unique global minimum.

The question now is how do we schedule the fixed point updates for efficiency? Since (5.23) is exactly a BP update, and (5.24) is exactly an IS update, the following is a common scheduling of (5.23, 5.24): alternately run a phase of BP (5.23) on the tree until convergence and perform a single scaling update (5.24). The schedule essentially implements the IS+BP procedure, except that BP is exact for a tree. Notice that during each BP phase we only need to either propagate messages towards the node where the next scaling update is going to occur (to compute the marginals required by the scaling update), or propagate messages away from the node where the previous scaling update occurred (to maintain marginalization consistency of the beliefs). Therefore each loop of the algorithm requires 1 scaling update, and $|E|$ propagation updates. However it is quite clear that this is not an efficient scheduling at all.

Let us illustrate this with an example. Consider the tree in figure 5.2.

Firstly suppose we have just run BP on the tree and performed a scaling update at node 1, and would now like to perform a scaling update at node 2. Before we do so we need to calculate the marginal $Q(x_2)$ at node 2. We can run another phase of BP on the tree to get $Q(x_2)$, but since the Lagrange multipliers at node 1 were the only ones altered since the last BP phase, the only messages relevant for calculating $Q(x_2)$ which have been altered are those lying on the path from node 1 to 2. In general, rather than running a full BP phase on the whole tree, we only need to perform BP updates from the node of the previous scaling update to the node of the next scaling update.

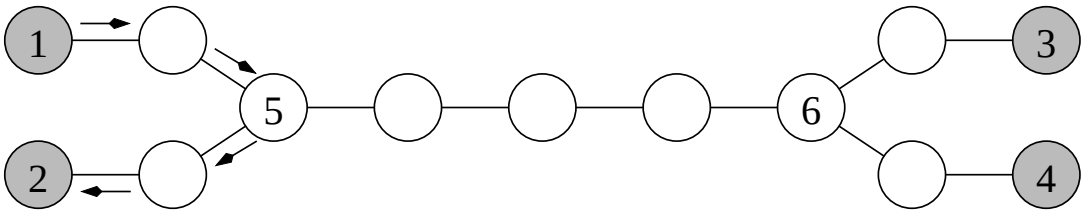


Figure 5.2: Scheduling propagation updates for UPS-T. The observed nodes are grey while the hidden nodes are white. The only propagation updates required between scaling updates at nodes 1 and 2 are those on the path from node 1 to node 2 (the arrows).

Secondly, we can improve IS+BP by scheduling the scaling updates appropriately. Consider again figure 5.2. Suppose we scheduled to perform scaling updates repeatedly on the nodes in the following order: 1, 3, 2, 4, 1, 3, 2, 4, ... This is an inefficient schedule because we had to run each BP update between nodes 5 and 6 twice for each loop through the scaling updates once. A better schedule would be to update nodes 1, 2, 3, 4 in order. In this case we only had to run each relevant BP update exactly once per loop.

In general, let the subtree $S \subset G$ consist of those edges in G which lay on the path between two nodes in V. S consists of exactly those edges in G on which propagation updates have to be performed in between scaling updates. Note that $V \subset V(S)$. Let $i_1 \in V$ be an observed node. We will perform updates in a depth first search manner on S starting at i_1. An example of this is illustrated in figure 5.3. Whenever a node in V is visited we will perform a scaling update at that node, and whenever an edge is traversed (either going down the tree or backtracking) the corresponding propagation update is performed. After the depth first search all nodes in V will have been visited. Further, in between any two scaling updates propagation updates are performed from the first node to the second. Hence the depth first search implements the efficient propagation schedule we were describing. Further, note that every relevant propagation update is performed exactly once only, hence it implements an efficient scheduling of the scaling updates as well.

The resulting algorithm, UPS-T, is summarized in algorithm 5.3. Each loop of UPS-T through steps 4 to 7 requires $|V|$ scaling updates and $2|E(S)|$ propagation updates. Steps 1 and

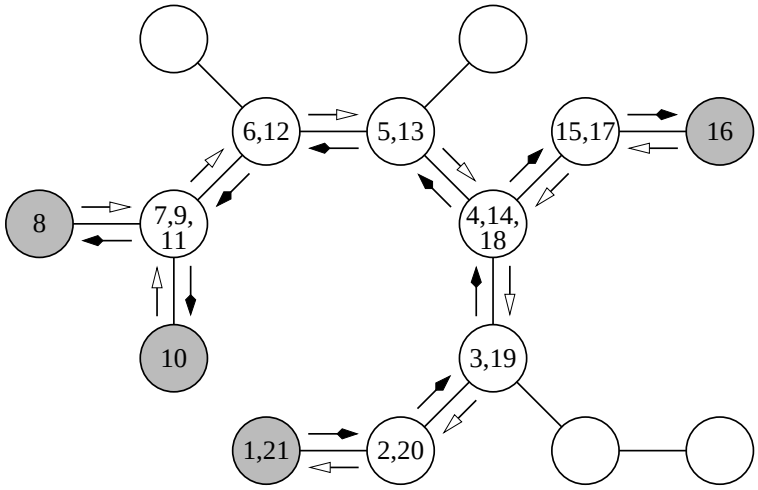


Figure 5.3: Scheduling scaling updates for UPS-T. Updates are performed in a depth first manner on the tree. The numbers describe the order in which nodes are visited. Black arrows are forward traversals and white arrows are backtracking traversals.

8 are required to make the beliefs consistent (i.e. to satisfy the MN constraints) since not all nodes in G are visited through steps 4 to 7. Each requires $2|E(G)|$ propagation updates.

5.6.3 Graphs with cycles

For graphs with cycles, $\mathcal{F}_{bethe}$ is neither exact nor convex. However we can make use of the fact that it is exact on trees to find a local minimum (or saddle point). The idea is that we clamp a number of hidden nodes to their current marginals such that the rest of the hidden nodes become singly connected, and apply UPS-T. Once UPS-T has converged, we clamp a different set of hidden nodes and apply UPS-T again. The algorithm can be understood as coordinate descent where we minimize $\mathcal{F}_{bethe}$ with respect to the unclamped nodes at each iteration.

Let $C \subset H$ be a set of clamped nodes such that every loop in the graph G contains a node from $U = V \cup C$. Define G' to be the graph obtained from G as follows. For each node $i \in U$ replicate it n_i times, and connect each replica to one neighbour of i and no other nodes. This is shown in figure 5.4b and 5.4c for the graph in figure 5.4a. Clearly G' will be singly connected. For $i', j' \in G'$ define $\psi_{i'j'}(x_{i'}, x_{j'}) = \psi_{ij}(x_i, x_j)$ where i and j are the original nodes in G.

Algorithm 5.3 UPS-T – Unified Propagation and Scaling on Trees.

1. Run propagation updates (5.23) until convergence.
2. Let $i_1 \in V$ be the root of depth first searches on S.
3. Run until convergence criterion is met:
4. Let $i_1, i_2, \ldots, i_s$ be the sequence of nodes visited during a depth first search starting at i_1. Note $i_s = i_1$ and $s = 2|E(S)| + 1$.
5. For $t = 1, 2, \ldots, s-1$:
6. If $i_t \in V$ perform scaling update (5.24) at i_t.
7. Perform propagation update (5.23) for $\lambda_{i_{t+1} i_t}(x_{i_t})$.
8. Run propagation updates (5.23) until convergence.

Similarly for $\psi_{i'}(x_{i'})$, $b_{i'j'}(x_{i'}, x_{j'})$ and $b_{i'}(x_{i'})$. Let $T \subset G'$ denote the trees in G'. Define

$$P'(x) = \prod_{T \subset G'} P'_T(x_T) \propto \prod_{T \subset G'} \prod_{(i'j') \in T} \psi_{i'j'}(x_{i'}, x_{j'}) \prod_{i' \in T} \psi_{i'}(x_{i'})^{1-n_{i'}} \tag{5.30}$$

$$Q'(x) = \prod_{T \subset G'} Q'_T(x_T) \propto \prod_{T \subset G'} \prod_{(i'j') \in T} b_{i'j'}(x_{i'}, x_{j'}) \prod_{i' \in T} b_{i'}(x_{i'})^{1-n_{i'}} \tag{5.31}$$

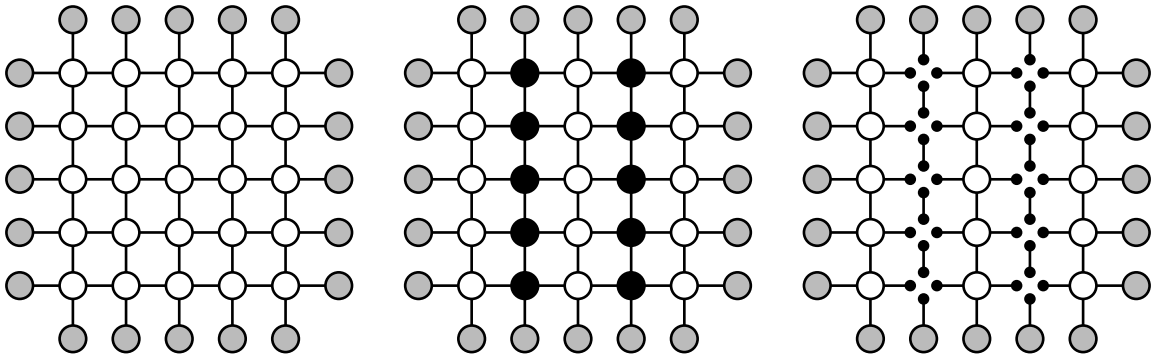
where $n_{i'}$ is the number of neighbours of node i' in G'. By regrouping terms in $\mathcal{F}_{bethe}$ we can show the following:

Theorem 5.6.1 *Let $\widehat{c}_i(x_i)$ be a distribution over x_i for $i \in C$. Then in the subspace defined by $b_i(x_i) = \widehat{c}_i(x_i)$ for $i \in C$ and by the MN and Obs constraints, we have*

$$\mathcal{F}_{bethe} = \sum_{T \subset G'} KL(Q'_T \| P'_T) + \sum_{i \in U} (1 - n_i) \sum_{x_i} \widehat{c}_i(x_i) \log \frac{\widehat{c}_i(x_i)}{\phi_i(x_i)} \tag{5.32}$$

To minimize $\mathcal{F}_{bethe}$, now all we have to do is to minimize each $KL(Q'_T \| P'_T)$ individually. We can already solve this using UPS-T. By clamping the marginals of nodes in C, we have reduced the problem to one solved by UPS-T, where the observed nodes are taken to include those in C. The overall algorithm is given in algorithm 5.4.

It is clear that $\mathcal{F}_{bethe}(Q^{(t)}) \leq \mathcal{F}_{bethe}(Q^{(t-1)})$ for all t. Now by using the fact that both scaling and propagation updates are fixed point equations for finding stationary points of $\mathcal{F}_{bethe}$ we have,



(a) (b) (c)

Figure 5.4: Clamping hidden nodes of a graph to make it singly connected. Hidden nodes are white; observed nodes are grey; and clamped nodes are black. The little nodes are the replicas.

Algorithm 5.4 UPS – Unified Propagation and Scaling

1. Initialize beliefs $Q^{(0)} = \{b_{ij}^{(0)}(x_i, x_j), b_i^{(0)}(x_i)\}$.
2. For $t = 1, 2, 3, \ldots$ until convergence criterion is met:
3. Find a set of nodes $C^{(t)}$ such that every loop in G is broken by $V \cup C^{(t)}$.
4. Using UPS-T, set $Q^{(t)} = \operatorname{argmin}\{\mathcal{F}_{bethe}(Q) \mid b_i(x_i) = b_i^{(t-1)}(x_i)$ for $i \in C^{(t)}$, and MN and Obs constraints are satisfied $\}$.

Theorem 5.6.2 *If for all t and $i \in H$ there is a $t_1 \geq t$ with $i \notin C^{(t_1)}$, then $Q^{(t)}$ will converge to a local minimum (or saddle point) of $\mathcal{F}_{bethe}$ with MN and Obs constraints satisfied.*

Notice that when the Obs constraints are delta functions and generalized inference reduces to ordinary inference, UPS reduces to a convergent alternative to loopy BP.

5.7 Experiments

In this section we report on two experiments on the feasibility of the various algorithms. In the first experiment we compared the speed of convergence of the various algorithms. In the second experiment we compared the accuracy of the two fastest algorithms – UPS and loopy IS.

In both experiments we used Boltzmann machines with 5×5 square lattice structure as shown in figure 5.4a. The observed nodes are on the edges of the lattice while the rest are hidden nodes. The states are $\{0, 1\}$ valued. The weights are sampled randomly from a Gaussian with mean 0 and standard deviation σ_w and the biases are sampled from a Gaussian with standard deviation σ_b and mean $-\sum\{$ incoming weights $\}/2$. The means of the biases are shifted so that if σ_b is small, the mean values of x_i will be approximately 0.5. In fact we can show by symmetry that if $\sigma_b = 0$ then the mean values of x_i are exactly 0.5. The desired observational marginals are $\widehat{o}_i(1) = 1/(1 + e^{\alpha_i})$ where α_i are sampled from a Gaussian with mean 0 and standard deviation σ_α. For different values of σ_α the distribution of observed marginals is quite different.

5.7.1 Speed of Convergence

We compared the speed of convergence for the following algorithms: loopy IS, IS+BP, GIS+BP (parallel GIS with marginals estimated by loopy BP), UPS-V (clamping columns of nodes every iteration as in figure 5.4b and UPS-HV (alternatingly clamping rows and columns). We tested the algorithms on 100 networks, with $\sigma_w = 5$, $\sigma_b = 1$ and $\sigma_\alpha = 4$. We find that the result is not sensitive to the settings of σ_w, σ_b and σ_α so long as the algorithms are able to converge without damping. The result is shown in figure 5.5. IS+BP and GIS+BP are slow because the loopy BP phase is expensive. UPS-V and UPS-HV both do better than IS+BP and GIS+BP because the inner loops are cheaper, and scaling updates on the Lagrange multipliers $\lambda_i(x_i)$ are run more frequently. Further we see that UPS-HV is faster than UPS-V since information is propagated faster throughout the network. Loopy IS is the fastest. However the next experiment shows that it also converges less frequently and there is a trade off between the speed of loopy IS and the stability of UPS.

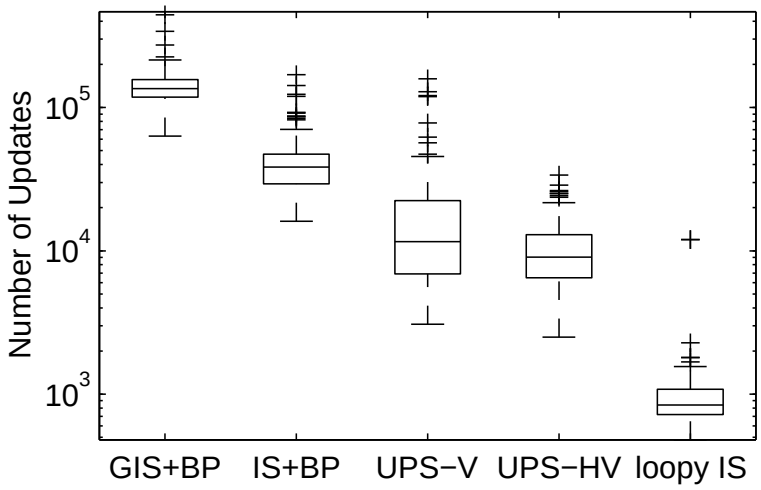


Figure 5.5: Speed of convergence of the various algorithms. The box lines are at the median and upper and lower quartiles, and the whiskers describe the extent of data. An algorithm or subroutine is considered converged if the beliefs change by less than 10^{-10}.

5.7.2 Accuracy of Estimated Marginals

We compared the accuracy of the estimated marginals obtained using UPS (in particular UPS-HV) and loopy IS for four possible types of constraints, as shown in figure 5.6.

In case (a), the constraint marginals are delta functions, so that generalized inference reduces down to ordinary inference, loopy IS reduces to loopy BP and UPS becomes a convergent alternative to loopy BP. In case (b), we did not enforce any Obs constraints so that the problem is one of estimating the marginals of the prior $P(x)$. The general trend is that loopy BP and UPS are comparable, and they perform worse as weights get larger, biases get smaller or there is less evidence. This confirms the results in (Welling and Teh, 2001). Further, we see that when loopy BP did not converge, UPS's estimates are not better than loopy BP's estimates. The reason is that UPS tends to converge to solutions where the marginal distributions are close to uniform when loopy BP did not converge. For a more detailed description of this see Welling and Teh (2001).

In cases (c) and (d) we set $\sigma_\alpha = 0.2, 2.0$, corresponding to $\widehat{o}_i(1) \approx 0.5$ and $\widehat{o}_i(1)$ spread out over $[0, 1]$ respectively. In these cases UPS and loopy IS did equally well when the latter

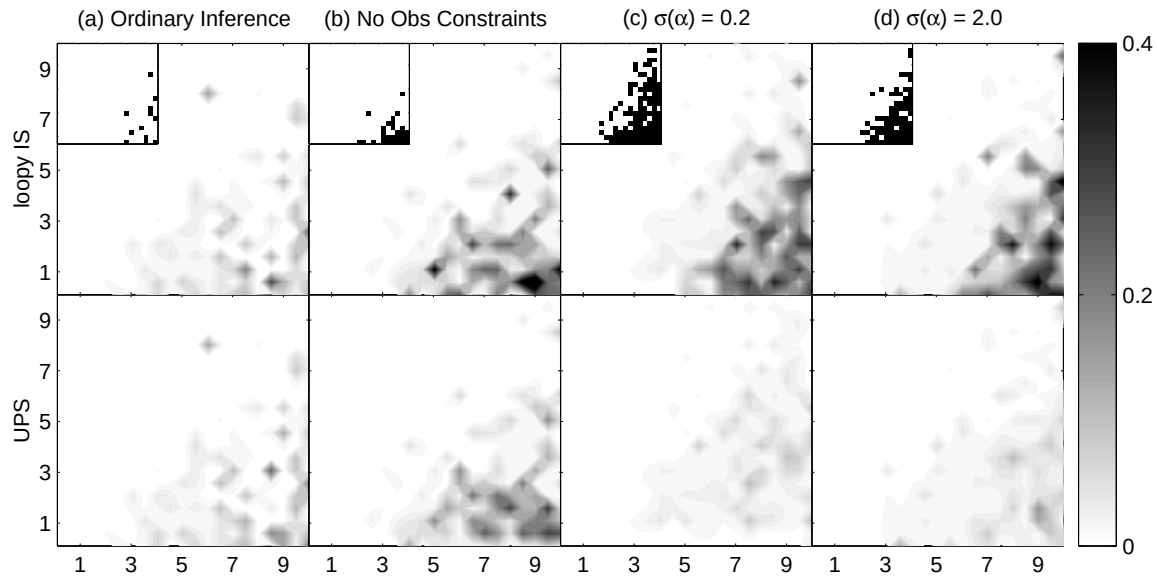


Figure 5.6: Each plot shows the mean absolute errors for various settings of σ_w (x-axis) and σ_b (y-axis). The top plots show errors for loopy IS and bottom plots show errors for UPS. The inset shows the cases (black) when loopy IS did not converge within 2000 iterations, with linear damping slowly increasing to 0.99. When loopy BP did not converge the errors are calculated from the current beliefs after 2000 iterations.

converged, but UPS continued to perform well even when loopy IS did not converge. Since loopy BP always converged when UPS performed well (for cases (a) and (b)), and we used very high damping, we conclude that loopy IS's failure to converge must be due to performing scaling updates before accurate marginals were available.

Concluding, we see that UPS is comparable to loopy IS when generalized inference reduces to ordinary inference, but in the presence of Obs constraints it is better.

5.8 Discussion

In this chapter we have shown that approximating the KL divergence with the Bethe free energy leads to viable algorithms for approximate generalized inference. We also find that there is an interesting and fruitful relationship between IS and loopy BP. Our novel algorithm UPS can

also be used as a convergent alternative to loopy BP for ordinary inference.

Interesting extensions to approximate generalized inference are to cluster nodes together to get more accurate approximations to the KL divergence analogous to the Kikuchi free energy, and to handle marginal constraints over subsets of nodes. We can also explore other algorithms to minimize $\mathcal{F}_{bethe}$, including the CCCP algorithm (Yuille, 2002).

In the next chapter, we will explore using the ideas developed in this section, in particular approximating the KL divergence with the Bethe free energy, and the close relationship between IS and loopy BP, to the problem of approximately learning undirected graphical models.

Acknowledgements and Remarks

An earlier version of this chapter first appeared in the Fifteenth International Conference on Neural Information Processing Systems (Teh and Welling, 2002). We would like to thank Jonathan Yedidia, Yair Weiss, Alan Yuille, Robert Cowell, Geoffrey Hinton and Zoubin Ghahramani for interesting discussions and suggestions. We thank the Gatsby Charitable Foundation for funding.

Chapter 6

The Bethe Free Energy for Learning

In the previous chapter, we showed how we can perform approximate inference by direct minimization of the Bethe free energy. An important concept there is to cast both loopy belief propagation (BP) and iterative scaling (IS) updates as fixed point equations of a single constrained optimization problem. In this chapter we extend the results derived in the previous chapter to learning undirected graphical models.

IS on junction trees is a standard algorithm for learning in undirected graphical models. By again combining the IS and BP updates into a single framework, we derive a more efficient message updating protocol than the well known effective IPF of Jiroušek and Přeučil (1995). When the junction tree has an intractably large maximum clique size we propose to maximize an approximate constrained entropy based on region graphs (Yedidia et al., 2002). Unfortunately, it is unclear how to generalize UPS to this case so we propose a "loopy" version of IPF to miximize the new objective instead. We show that this yields accurate estimates of the weights of undirected graphical models in a simple experiment.

6.1 Introduction

Junction trees are widely used as efficient representations for probability models defined on graphs. For instance, to perform exact inference in Bayesian networks one typically transforms

the directed graph into a junction tree and computes the posterior probability over the cliques of the junction tree using local propagation rules. Two out of many well known schemes for this purpose are Hugin propagation (Jensen, 1996) and Shafer-Shenoy propagation (Shafer and Shenoy, 1990).

Junction trees are also indispensable for learning graphical models from data through the *iterative proportional fitting* (IPF) procedure, otherwise known as *iterative scaling* (IS) (Jiroušek and Přeučil, 1995). This *effective IPF* procedure represents the joint probability distribution in terms of the clique marginals of the junction tree, and alternates between updating the parameters of the model using IPF and propagating that change to the rest of the model using the junction tree. For structured problems, the junction tree representation reduces the space and time complexity of the IPF procedure drastically.

The first result we present in this chapter is a further decrease in the time complexity of the effective IPF procedure. It is shown that both the IPF and junction tree propagation updates are fixed point equations of a maximum entropy problem with certain constraints. This unifying view lifts the strict separation in the effective IPF procedure between IPF and junction tree propagation, and allows for more efficient schedulings of the IPF and junction tree propagation updates.

For some graphs the maximum clique size in the corresponding junction tree is still intractably large, and the problem needs to be tackled through approximations. To this extent we propose a framework closely related to an exciting recent technique for approximate inference variously known as loopy belief propagation, sum-product algorithm, or generalized distributive law (Yedidia et al., 2002). Using knowledge of the close relationship between propagation and IPF updates, we propose a procedure that performs approximate IPF on *region graphs*, which are natural extensions of junction trees that may contain cycles and be designed to have smaller clique sizes. This *loopy IS* procedure consists of running fixed point equations which solve for stationary points of a constrained approximate entropy similar to the region graph free energies.

In section 6.2 we describe the maximum entropy problem that is the focus of the chapter,

as well as the classical iterative scaling algorithm. We also show the relationship between maximum entropy and maximum likelihood learning of graphical models. In section 6.3 we describe the effective IPF procedure. Section 6.4 then describes the unifying view, as well as our efficient schedule. Section 6.5 deals with approximate IPF on region graphs, and section 6.6 shows in a simple experiment the efficacy of the approximation. Section 6.7 closes with some discussion and extensions.

6.2 Maximum Entropy

In this section we review the maximum entropy framework and its relationship to maximum likelihood learning in undirected graphical models. Let N be a set of nodes. Each node $i \in N$ is associated with a variable X_i. Denote the finite domain of values of X_i by $\mathbb{X}_i$ and let $x_i \in \mathbb{X}_i$ be a value of X_i. For a set of nodes $n \subset N$ let $X_n = (X_i)_{i \in n}$ be the variable associated with the nodes in n, $\mathbb{X}_n = \prod_{i \in n} \mathbb{X}_i$ be the domain of X_n, and $x_n \in \mathbb{X}_n$ be values of X_n. For simplicity we write $X_N = X$ and $X_{\setminus n} = X_{N \setminus n}$; similarly for x and $\mathbb{X}$.

Let A be a family of subsets (clusters) of N. On each cluster $\alpha \in A$ we are given a joint distribution $\widehat{p}_\alpha(x_\alpha)$ over the random variables X_α[1]. The family of distributions $\{\widehat{p}_\alpha\}_{\alpha \in A}$ is consistent if there is a distribution $P(x)$ satisfying the marginals $P(x_\alpha) = \widehat{p}_\alpha(x_\alpha)$ for all $\alpha \in A$. In this chapter we assume that $\{\widehat{p}_\alpha\}$ is indeed consistent. In such a case let the maximum entropy extension be

$$\operatorname*{argmax}_P \left\{ H(P) \;\middle|\; P(x_\alpha) = \widehat{p}_\alpha(x_\alpha) \, \forall \alpha \in A \right\} \tag{6.1}$$

where the entropy is $H(P) = -\sum_x P(x) \log P(x)$ and the domain of maximization is over the probability simplex.

We use Lagrange multipliers $\lambda_\alpha(x_\alpha)$ to impose the marginal constraints and γ to enforce

[1]The extension to being given feature expectations $\widehat{f}_\alpha = E[f_\alpha(x)]$ is straight-forward and described in section 6.7.

normalization ($\sum_x P(x) = 1$). The Lagrangian is

$$\mathcal{L} = H(P) - \sum_{\alpha, x_\alpha} \lambda_\alpha(x_\alpha)\Big(\widehat{p}_\alpha(x_\alpha) - \textstyle\sum_{x_{\setminus\alpha}} P(x)\Big) - \gamma\Big(1 - \textstyle\sum_x P(x)\Big) \tag{6.2}$$

Zeroing derivatives of $\mathcal{L}$ with respect to $P(x)$ and γ,

$$P(x) = e^{\sum_\alpha \lambda_\alpha(x_\alpha) + \gamma - 1} \tag{6.3}$$

$$\gamma = 1 - \log \sum_x e^{\sum_\alpha \lambda_\alpha(x_\alpha)} \tag{6.4}$$

This expresses the primal variables $P(x)$ in terms of the dual variables $\lambda_\alpha(x_\alpha)$. Finally, to solve for $\lambda_\alpha(x_\alpha)$, we substitute (6.3, 6.4) into (6.2) to obtain the dual cost

$$\mathcal{L}' = -\sum_{\alpha, x_\alpha} \lambda_\alpha(x_\alpha)\widehat{p}_\alpha(x_\alpha) + \log \sum_x e^{\sum_\alpha \lambda_\alpha(x_\alpha)} \tag{6.5}$$

Because the original cost function $H(P)$ is concave, its maximum coincides with the minimum of the dual cost function, and the maximum entropy extension is given in terms of the optimal $\lambda_\alpha(x_\alpha)$. Now the dual cost function $\mathcal{L}'$ is convex and can be solved by coordinate-wise descent in $\lambda_\alpha(x_\alpha)$. This is the classical iterative scaling algorithm (Deming and Stephan, 1940), given by the following updates:

$$\lambda_\alpha(x_\alpha) \leftarrow \lambda_\alpha(x_\alpha) + \log \frac{\widehat{p}_\alpha(x_\alpha)}{P(x_\alpha)} \tag{6.6}$$

where $P(x)$ is given by (6.3, 6.4). In terms of the primal variables $P(x)$, we can understand each update of (6.6) as setting the marginal $P(x_\alpha)$ to be $\widehat{p}_\alpha(x_\alpha)$. In fact, (6.6) is equivalent to the following primal update:

$$P(x) \leftarrow P(x)\frac{\widehat{p}_\alpha(x_\alpha)}{P(x_\alpha)} \tag{6.7}$$

The maximum entropy framework is intimately related to maximum likelihood learning of undirected graphical models (Della Pietra et al., 1997). Let the clusters of the graphical model be given by A. The distribution expressed by the graphical model has the form

$$P(x) = \frac{1}{Z}\exp\left(\textstyle\sum_\alpha \lambda_\alpha(x_\alpha)\right) \tag{6.8}$$

where $\lambda_\alpha(x_\alpha)$ are the parameters of the model and Z is the normalizing partition function. Let $\widehat{p}(x)$ be an empirical distribution obtained from a set of fully observed training data. The average log likelihood of the data is then

$$\sum_x \widehat{p}(x) \log P(x) = \sum_\alpha \widehat{p}(x_\alpha)\lambda_\alpha(x_\alpha) - \log Z \tag{6.9}$$

which is easily seen to be the negative of the maximum entropy dual cost function (6.5). Further, the distribution (6.8) is equivalent to (6.3, 6.4), hence the form of the graphical model can be derived from maximum entropy considerations with marginal constraints. Note that in this case the given distributions $\widehat{p}_\alpha(x_\alpha)$ are simply the marginal distributions $\widehat{p}(x_\alpha)$ of the empirical distribution, hence they are always consistent.

6.3 Junction Trees

The straight forward implementation of iterative scaling uses a simple probability table to represent $P(x)$. As $|\mathbb{X}|$ grows exponentially with $|N|$ the computational cost of the algorithm is high: each iterative scaling update requires $O(|\mathbb{X}|)$ time and $O(|\mathbb{X}|)$ space. In this section we review the effective IPF procedure of Jiroušek and Přeučil (1995) which reduces the computational costs drastically for structured domains.

In short, effective IPF uses a junction tree to represent $P(x)$ instead of a simple probability table. At each step of iterative scaling, our distribution P is given by (6.8), which has the structure of an undirected graphical model with nodes N and clusters A. The corresponding graph has an edge between two nodes if both are in the same cluster $\alpha \in A$. After triangulating this graph, the maximal cliques C form a junction tree with separators S separating them. By construction, each cluster $\alpha \in A$ is contained in some maximal clique $c \in C$, therefore $P(x)$ is decomposable with respect to the junction tree, i.e.

$$P(x) = \frac{\prod_{c\in C} P(x_c)}{\prod_{s\in S} P(x_s)} \tag{6.10}$$

Rather than representing $P(x)$ as a straight probability table, we represent it as a set of smaller

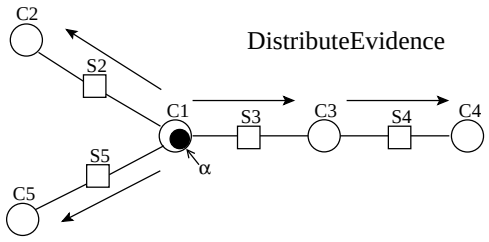


Figure 6.1: An ordering satisfying the running intersection property to distribute the iterative scaling change at c_1 to the rest of the graph.

tables $\{P_c(x_c)\}_{c\in C}$ on the cliques[2]. These tables have to be consistent, i.e. if c_1, c_2 are neighbouring cliques with s separating them, then $P_{c_1}(x_s) = P_{c_2}(x_s)$[3].

Consider the primal iterative scaling update (6.7). Let $c_1 \in C$ be a clique containing α. The iterative scaling update can be performed on c_1 rather than over all N:

$$P_{c_1}(x_{c_1}) \leftarrow P_{c_1}(x_{c_1})\frac{\widehat{p}_\alpha(x_\alpha)}{P_{c_1}(x_\alpha)} \tag{6.11}$$

This changes the distribution $P_{c_1}(x_{c_1})$ and makes it inconsistent with the other tables. To maintain consistency, we propagate this change to the rest of the junction tree using a standard DistributeEvidence phase[4]. This is illustrated in figure 6.1. Let $c_1, c_2, \ldots$ be an ordering of the cliques satisfying the *running intersection property*: for each t there is a unique $\sigma(t) < t$ with $s_t \doteq c_t \cap c_{\sigma(t)} = c_t \cap (\cup_{t'<t} c_{t'})$. DistributeEvidence then amounts to

$$P_{c_t}(x_{c_t}) \leftarrow P_{c_t}(x_{c_t})\frac{P_{c_{\sigma(t)}}(x_{s_t})}{P_{c_t}(x_{s_t})} \tag{6.12}$$

for $t = 2, 3, \ldots$. In essence, we are replacing the marginal $P_{c_t}(x_{s_t})$ with the new marginal $P_{c_{\sigma(t)}}(x_{s_t})$ and the information carried in the marginals flows outward from the original cluster α.

When the cliques are relatively small, the junction tree representation of $P(x)$ is much more efficient than a straight probability table. Let $M = \max_{c\in C} |\mathbb{X}_c| \ll |\mathbb{X}|$. Each iterative scaling

[2]The tables on the separators are not required as they can be computed by marginalizing a neighbouring clique.

[3]Because the cliques form a tree, this *local* consistency is equivalent to the *global* consistency we encountered for $\{\widehat{p}_\alpha(x_\alpha)\}$. This is unfortunately not true if the cliques do not form a tree (see section 6.5).

[4]Equivalently, we can use a CollectEvidence phase *before* each iterative scaling update to compute the required marginal $P_{c_1}(x_{c_1})$ (Bach and Jordan, 2002).

update is followed by $|S|$ propagation updates. So both the time and storage requirements are $O(|S|M)$ per iterative scaling update.

6.4 Unifying Propagation and Scaling

In section 6.3 we introduced junction trees as simply a computational tool to improve the efficiency of the iterative scaling procedure. We will show in this section that both the propagation updates (6.12) and the iterative scaling updates (6.11) can be derived as fixed point equations of a constrained maximization problem. A consequence of this is that any intermixed schedule of iterative scaling and junction tree propagation updates will converge to the maximum entropy solution. This allows us more flexibility in designing efficient schedules of the updates. In particular, we propose a new scheduling which requires only $2|S|$ propagation updates to perform all $|A|$ iterative scaling updates once. This is more efficient than the algorithm in section 6.3.

6.4.1 Constrained Maximization

Consider the following constrained maximization problem:

$$\underset{\{P_c,P_s\}}{\operatorname{argmax}} \Big\{ \sum_c H(P_c) - \sum_s H(P_s) \;\Big|\; P_c(x_\alpha) = \widehat{p}_\alpha(x_\alpha), \\ P_c(x_s) = P_s(x_s)\, \forall \alpha, s, c \text{ with } \alpha, s \subset c \Big\} \quad (6.13)$$

where the domains of P_c and P_s are probability simplexes. When the constraints are satisfied, the distributions on the cliques and separators are consistent hence they can be combined into a single distribution $P(x)$ using (6.10). Then the cost function is

$$\sum_c H(P_c) - \sum_s H(P_s) = H(P) \quad (6.14)$$

This means that (6.13) is a specific case of the original maximum entropy problem (6.1) where P is assumed decomposable with respect to the junction tree. But section 6.3 shows that the maximum entropy extension is itself decomposable with respect to the same junction tree.

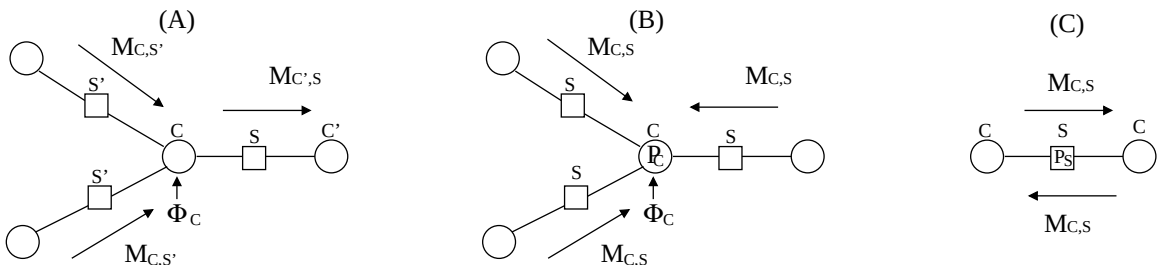


Figure 6.2: (a) Shafer-Shenoy propagation updates. (b) Computing clique distributions from messages. (c) Computing separator distributions from messages.

Hence the marginal distributions of the maximum entropy extension form a solution to (6.13) and the two problems are actually equivalent.

Again we will use Lagrange multipliers to solve (6.13). Let $\lambda_{cs}(x_s)$ impose the marginal consistencies and let γ_c, γ_s make sure P_c and P_s are normalized. We identify each $\alpha \in A$ with a clique $c_\alpha \in C$ and let $\lambda_\alpha(x_\alpha)$ impose the given constraint that $P_{c_\alpha}(x_\alpha) = \widehat{p}_\alpha(x_\alpha)$. Let $A_c = \{\alpha \in A | c_\alpha = c\}$. The Lagrangian is then

$$\begin{aligned}\mathcal{L} = &\sum_c H(P_c) - \sum_s H(P_s) - \sum_{v \in S \cup C} \gamma_v \left(\textstyle\sum_{x_v} P_v(x_v) - 1 \right) \\ &- \sum_{c,s,x_s} \lambda_{cs}(x_s) \left(P_s(x_s) - \textstyle\sum_{x_{c \setminus s}} P_c(x_c) \right) \\ &- \sum_{\alpha, x_\alpha} \lambda_\alpha(x_\alpha) \left(\widehat{p}_\alpha(x_\alpha) - \textstyle\sum_{x_{c_\alpha \setminus \alpha}} P_{c_\alpha}(x_{c_\alpha}) \right) \end{aligned} \tag{6.15}$$

Solving the Lagrangian as before, we find that the marginal distributions are

$$P_c(x_c) \propto e^{\sum_s \lambda_{cs}(x_s) + \sum_{\alpha \in A_c} \lambda_\alpha(x_\alpha)} \tag{6.16}$$

$$P_s(x_s) \propto e^{\sum_c \lambda_{cs}(x_s)} \tag{6.17}$$

while λ_α and λ_{cs} are updated with the fixed point equations

$$\lambda_\alpha(x_\alpha) \leftarrow \lambda_\alpha(x_\alpha) + \log \frac{\widehat{p}_\alpha(x_\alpha)}{P_{c_\alpha}(x_\alpha)} \tag{6.18}$$

$$e^{\lambda_{c's}(x_s)} \leftarrow \propto \sum_{x_{c \setminus s}} e^{\sum_{s' \neq s} \lambda_{cs'}(x_{s'}) + \sum_\alpha \lambda_{c\alpha}(x_\alpha)} \tag{6.19}$$

where c' and c are the two cliques separated by s, and s' are other separators neighbouring c. We see that iterative scaling updates (6.6) are fixed point equations (6.18) to solve the Lagrangian.

Also identifying messages and potentials as

$$M_{cs}(x_s) \doteq e^{\lambda_{cs}(x_s)} \qquad \phi_c(x_c) \doteq e^{\sum_\alpha \lambda_{c\alpha}(x_\alpha)} \tag{6.20}$$

(6.19) is easily shown to be

$$M_{c's}(x_s) \leftarrow\propto \sum_{x_{c\setminus s}} \phi_c(x_c) \prod_{s' \neq s} M_{cs'}(x_{s'}) \tag{6.21}$$

which can be identified as a Shafer-Shenoy propagation update for junction trees (Shafer and Shenoy, 1990). The marginal distributions are then given by

$$P_c(x_c) \propto \phi_c(x_c) \prod_s M_{cs}(x_s) \tag{6.22}$$

$$P_s(x_s) \propto \prod_c M_{cs}(x_s) \tag{6.23}$$

The Shafer-Shenoy updates are depicted in figure 6.2.

Sometimes, it is more intuitive and effective to perform iterative scaling using the primal updates of (6.7) rather than the dual updates of (6.6). Similarly, sometimes propagation updates which deal directly with clique marginals are more desirable. One of these is Hugin propagation, given by

$$P_{c'}(x_{c'}) \leftarrow P_{c'}(x_{c'}) \frac{P_c(x_s)}{P_s(x_s)} \qquad P_s(x_s) \leftarrow P_c(x_s) \tag{6.24}$$

Hugin propagation can be shown to be equivalent to Shafer-Shenoy where we identify the messages and marginal distributions using (6.22, 6.23).

6.4.2 Efficient Scheduling

The previous subsection shows that both iterative scaling and Shafer-Shenoy propagation updates are fixed point equations to solve the maximum entropy problem (6.13). Because the cost function of (6.13) is concave in the space where the constraints are satisfied, the fixed point equations are guaranteed to converge to the global optimum. However this does not imply anything about the efficiency of various schedules. We now propose a particular class of schedules which will be efficient in the sense to be defined below.

We can understand the iterative scaling update (6.18) as changing the Lagrange multiplier $\lambda_{c\alpha}(x_\alpha)$, given the current estimate $P_c(x_c)$ of the true marginal distribution $P(x_c)$, so as to satisfy the constraint $P(x_\alpha) = \widehat{p}_\alpha(x_\alpha)$. On the other hand, the propagation updates (6.19) or (6.21) compute the required marginal distributions $P(x_c)$ and store them in $P_c(x_c)$ from the current Lagrange multipliers. If the propagation updates are run until convergence after every iterative scaling update, then the given $P_c(x_c)$ will be the true $P(x_c)$. This is the schedule of the effective IPF procedure. However it is clear that this schedule is inefficient since it calculates all the marginal distributions exactly even though only one is needed for the next iterative scaling update. On the other hand, if the propagation updates have not converged before an iterative scaling update is performed, then the calculated marginal distribution $P_c(x_c)$ might not be exact. As a result the iterative scaling update might not be as effective.

In view of the above issues, we shall show that our proposed schedule, unified propagation and scaling for junction trees (UPS-JT), is efficient in that it satisfies the following properties: (1) whenever an iterative scaling update is performed, the current estimate of the required marginal $P_c(x_c)$ is exact; (2) between any two iterative scaling updates, only at most one propagation update is performed to ensure $P_c(x_c)$ is exact for the second iterative scaling update.

Algorithm 6.1 UPS-JT – Unified Propagation and Scaling for Junction Trees

1. Initialize the junction tree so that each P_c and P_s is uniform.
2. Initialize messages or Lagrange multipliers to uniform as well.
3. Initialize c_1 to some clique in C.
4. For $t = 1, 2, \ldots$ until convergence criterion is met:
5. Perform iterative scaling updates for those clusters $\alpha \in A_{c_t}$ identified with c_t.
6. Choose a clique c_{t+1} neighbouring c_t.
7. Perform the propagation update from c_t to c_{t+1}.

Note that UPS-JT does not prescribe the ordering in which we visit the cliques except for the implicit requirement that all cliques are visited enough times. One possible ordering is to

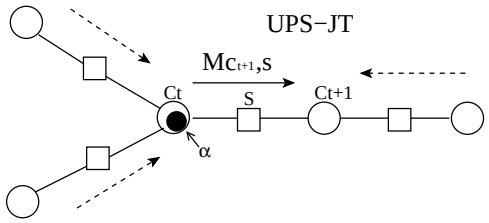


Figure 6.3: The dashed lines are the messages which are still correct, while the solid line denotes the message that is updated to become correct.

visit the cliques in a depth first search manner on the junction tree. This guarantees that every Lagrange multiplier is updated once for a total of $2|S|$ propagation updates. This is much more efficient than the ordering in the effective IPF procedure (Jiroušek and Přeučil, 1995), which takes $|S|$ propagation updates for every iterative scaling update. The UPS-T algorithm of chapter 5 uses this scheduling.

Compared with the effective IPF procedure, we see that UPS-JT performs multiple iterative scaling updates on each clique, and, more importantly, substituted a full DistributeEvidence phase with a single propagation update. Hence it satisfies condition 2 above. We now prove by induction that condition 1 is satisfied when using Shafer-Shenoy propagation. The inductive hypothesis is that at each iteration all incoming messages to c_t are correct. At time $t = 1$ this is trivially true. In figure 6.3 we depict one step of the UPS-JT algorithm for time $t \geq 1$. we have updated the Lagrange multipliers $\lambda_\alpha(x_\alpha)$ in clique c_t. Next, as required by step 7 of the UPS-JT algorithm, we choose a neighboring clique in the tree, c_{t+1}, to perform our next scaling update. As required by condition 1, we need the exact marginal $P(x_{c_{t+1}})$. To compute that we collect evidence to the clique c_{t+1}, by propagating inward from the leaves. But note that all messages (e.g. dashed arrows in figure 6.3), except message $M_{c_{t+1}s}(x_s)$ (solid arrow), are unchanged by the scaling update at node c_t. Hence, we only recompute the latter and use (6.16) to get the correct marginal at clique c_{t+1}.

6.5 Loopy Iterative Scaling

UPS-JT is an efficient algorithm for entropy maximization if the cliques of the junction tree are small. However, its complexity scales exponentially with the size of the maximal clique. To combat this, we propose an approximate algorithm named *loopy iterative scaling* based on *region graphs*.

6.5.1 Region Graphs

After Yedidia et al. (2002) we define a region graph as an acyclic directed graph where the vertices are labelled with subsets of nodes, or regions. The top layer consists of a family of large regions which cover the original graph, such that each cluster $\alpha \in A$ is contained in at least one of them. A directed edge can only exist between a parent and a child if the nodes associated with the child are a subset of the nodes associated with its parent. With each vertex (or region) we will also associate a "counting number" c_r,

$$c_r = 1 - \sum_{r' \in \mathrm{Super}(r)} c_{r'} \tag{6.25}$$

where $\mathrm{Super}(r)$ consists of all regions which strictly contain r, and $c_r = 1$ for the top layer regions. A valid region graph should fulfill the following two conditions: (1) for each node, the subgraph induced by the vertices containing that node must be connected; (2) the sum of the counting numbers of each such subgraph must add up to one.

With each region graph we can associate an approximate entropy

$$\mathcal{H}(\{P_r\}) = \sum_r c_r \, H(P_r) \approx H(P). \tag{6.26}$$

and generalized belief propagation algorithms to maximize it (see Yedidia et al. (2002) for details). The collection $\{P_r\}$ are now *approximate* marginals satisfying *local* consistency constraints[5]: $P_r(x_s) = P_s(x_s)$ for every child s of region r. The region-based entropy maximiza-

[5]Note that since the region graph can contain cycles there might not be a distribution which is consistent with all $\{P_r\}$, i.e. they may not be *globally* consistent.

tion problem is then given by

$$\underset{\{P_r\}}{\operatorname{argmax}} \left\{ \mathcal{H}(\{P_r\}) \;\middle|\; P_r(x_\alpha) = \widehat{p}_\alpha(x_\alpha), P_r(x_s) = P_s(x_s) \,\forall \alpha, s, r \text{ with } \alpha, s \subset r \right\} \tag{6.27}$$

The solutions to (6.27) are approximations to the marginals of the true maximum entropy distribution. Further, the optimal Lagrange multipliers are approximate solutions to the maximum likelihood parameters, if we are doing maximum likelihood training of a graphical model.

Solving the above region-based entropy maximization problem, it is not hard to show that the fixed point equations are given again by the scaling updates (6.18) for the Lagrange multipliers, and the fixed point equations of generalized belief propagation. Here the Lagrange multipliers associated with the constraints $P_{r_\alpha}(x_\alpha) = \widehat{p}_\alpha(x_\alpha)$ are $\lambda_{r_\alpha}(x_\alpha)$, and the potentials for the top layer regions r are $\phi_r(x_r) \doteq e^{\sum_{\alpha \in A_r} \lambda_{r\alpha}(x_\alpha)}$, where r_α is some top layer region containing α, and $A_r = \{\alpha | r_\alpha = r\}$. These *loopy iterative scaling* updates can be performed using any convenient scheduling, but convergence is unfortunately not guaranteed.

Unfortunately, the technique of clamping units used in chapter 5 to derive convergent alternatives to loopy iterative scaling cannot be extended to here. The reason is that in chapter 5 the observational constraints and the clamping constraints are always consistent, while adding constraints by clamping the marginals at some subset of nodes here can easily make the constraints inconsistent. However it is conceivable that algorithms which maximize lower bounds on the approximate entropy (e.g. Yuille, 2002) can be made to work here.

Although region graphs conveniently translate into loopy iterative scaling algorithms, it has not been made clear how to construct a valid region graph given a family of large regions. There are two distinct methods described in the literature, one based on junction graphs, the other called the cluster variation method.

6.5.2 Junction Graph Method

A junction graph is a two layer region graph with large regions called cliques and their children, called separators. Since the region graph has such a simple structure, we typically ignore the directionality of the edges. For each node we require that the subgraph constructed from all

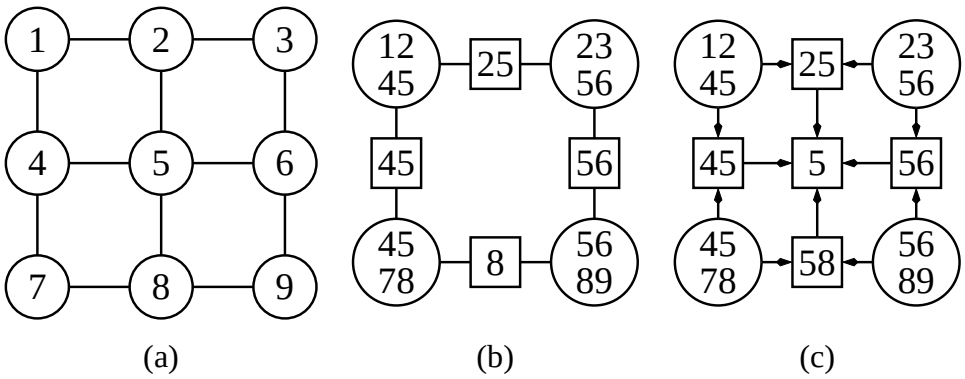


Figure 6.4: (a) An example of a graphical model. (b) A junction graph for the model. Notice that the separator on the bottom only contains 8 and not 5. (c) A region graph constructed using the cluster variational method.

cliques and separators containing that node should form an undirected tree. Note that this condition is stronger than the region graph condition, and automatically ensures that all counting numbers in that subgraph sum to one. This property is the equivalent of the running intersection property for junction trees and guarantees that junction graphs "look like" junction trees locally. An example of a junction graph is shown in figure 6.4.

Given a cluster set A, there is a variety of junction graphs that are consistent with A. On one end of the spectrum, there are junction trees, on which we can perform exact entropy maximization. On the other end, we have junction graphs with small cliques that are poor approximations but admit efficient algorithms to maximize the approximate entropy. In fact, Aji and McEliece (2001) show that for any collection of subsets of nodes (in particular, for A) it is easy to construct a junction graph whose cliques consist precisely of the subsets in the collection: first define the separators as the intersections of pairs of cliques, then for every node construct the subgraph induced by the cliques and separators containing that node and delete nodes from separators as well as remove separators which are empty until the subgraph is a tree.

Junction graphs are particularly convenient because the propagation updates reduce precisely to the junction tree updates (6.19). Moreover, the approximate entropy (6.26) reduces to

the following *Bethe entropy*,

$$\mathcal{H}(\{P_c, P_s\}) = \sum_c H(P_c) - \sum_s H(P_s) \tag{6.28}$$

6.5.3 Cluster Variation Method

An alternative road to constructing valid region graphs is provided by the cluster variation method. We start again with a family of large regions such that each cluster is contained in at least one of them. Next, the children of the large regions are defined as their intersections[6], and the children of those are given by the intersections of the intersections. This process is repeated until no further (non-trivial) region can be added. The resultant layered region graph, with counting numbers assigned to regions using (6.25) is automatically valid, and the corresponding approximate entropy is known as the Kikuchi entropy. The corresponding loopy iterative scaling algorithm is now based on the generalized belief propagation algorithms as described in Yedidia et al. (2001).

6.6 Experiment

We explored the behaviour of loopy iterative scaling on junction graphs on a simple task of learning the weights of pairwise Markov networks. The training data consists of 8×8 binary images of hand-written '8's pre-processed from the CEDAR dataset (see figure 6.5).

First we generated the models to be fit to the data. These are generated so that they could assign reasonably high likelihood to the data, while at the same time have junction trees of manageable sizes. To do this, we first computed the maximum likelihood tree. This is obtained by calculating the mutual information from the training data between all pairs of pixels. Then the maximum likelihood tree is the maximum weight spanning tree where the weights are the mutual informations. Given the maximum likelihood tree, the models are generated by randomly sampling 5% of the edges connecting nodes a distance of at most 6 apart on the

[6]For each layer we do not include regions which are subsets of other regions in that layer.

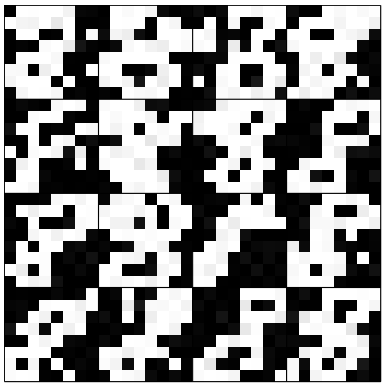


Figure 6.5: Some examples of the images of '8's. The images have been binarized by thresholding.

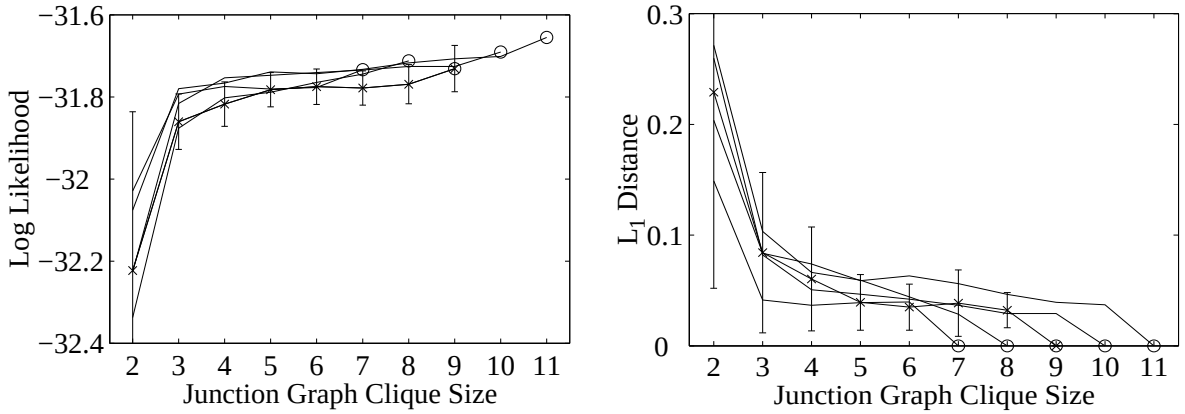
maximum likelihood tree. Junction trees are constructed by triangulating the resulting graphs using node elimination, where at each stage the node with the least neighbors is chosen. The maximal clique sizes of these junction trees vary between 7 and 11.

Second we constructed the junction graphs over which we approximate the entropy. For each model, starting with the edges as the cliques of the junction graph, a range of junction graphs is constructed by growing its cliques one node at a time, and making sure that the resulting cliques are contained within the cliques of the corresponding junction tree. At each stage a node is added to a clique such that the increase in the total mutual information[7] is largest, and removing cliques which are subsumed by other cliques.

Finally, on all the junction graphs so constructed we used loopy iterative scaling to learn the parameters, where both scaling and propagation updates were damped geometrically. Further, to encourage convergence, propagation updates were iterated until convergence before the scaling updates were performed once. Using these convergence improving measures we find that all our loopy iterative scaling runs converge (although we note that loopy iterative scaling is not guaranteed to converge).

After loopy iterative scaling has converged, we assessed the accuracy of the results using

[7]Total mutual information is defined as the sum of the mutual informations between all pairs of nodes within the same cliques.



(a) (b)

Figure 6.6: (a) Average log likelihoods over training data as a function of junction *graph* clique size. Each curve is averaged over all models with a certain maximal clique size of the corresponding junction *tree*. For models of treewidth 9, we also plotted the standard deviation of the log likelihoods. The circles denote the log likelihood under the exact maximum likelihood parameters (i.e. when the junction graph coincides with the junction tree). (b) Average L_1 distances of the various models. The notations are the same as in (a).

two measures. In figure 6.6a we plot the log likelihoods of the data under the learned models, while in figure 6.6b we show the L_1 distances between the empirical marginal distributions $\widehat{p}_\alpha(x_\alpha)$ and those of the learned models $P(x_\alpha)$, averaged over the edges α. The approximation is reasonably accurate, and its accuracy improves as we increase the maximum clique size of the junction graph.

6.7 Discussion

In this chapter we have shown that propagation and iterative scaling on junction trees can be unified as fixed point equations for solving a certain constrained maximum entropy problem. From this insight we have proposed a more efficient scheduling for iterative scaling on junction trees. For graphs with a maximal clique size which is prohibitively large, we have proposed a

loopy iterative scaling algorithm, based on region graphs.

There are a number of important extensions of the methods discussed in this chapter. Firstly, rather than finding a distribution with maximum entropy we can find a distribution with minimum relative entropy (KL divergence) to a given distribution P_0. The relative entropy from P to P_0 is defined to be

$$KL(P\|P_0) = \sum_x P(x) \log \frac{P(x)}{P_0(x)} \tag{6.29}$$

When P_0 is the uniform distribution, minimizing $KL(P\|P_0)$ is equivalent to maximizing the entropy of P. This extension is useful when we would like to learn undirected graphical models where certain potentials are fixed. These potentials are cast into P_0 in the minimum relative entropy framework. Our results on unifying junction tree propagation and iterative scaling carry through if P_0 is decomposable with respect to the junction tree as well.

Secondly, the results are straightforwardly extended to constraints on feature expectations, rather than on marginals. Here each cluster $\alpha \in A$ is associated with a vector-valued feature $f_\alpha : \mathbb{X}_\alpha \to \mathbb{R}^{d_\alpha}$, and $P(x)$ is constrained to have $\sum_x P(x) f_\alpha(x_\alpha) = \widehat{f_\alpha} \in \mathbb{R}^{d_\alpha}$. When dualized, the maximum entropy problem becomes a maximum likelihood problem for an undirected graphical model where f_α are the features. The Lagrange multipliers $\lambda_\alpha \in \mathbb{R}^{d_\alpha}$ imposing the expectation constraints become the weights corresponding to the features in the undirected graphical model

$$P(x) = \frac{1}{Z} e^{\sum_\alpha \lambda_\alpha^T f_\alpha(x_\alpha)} \tag{6.30}$$

The required expectations $\widehat{f_\alpha}$ are obtained by averaging over a training set. The algorithms discussed in this chapter therefore open the way for more efficient training of maximum entropy models (Della Pietra et al., 1997), conditional maximum entropy models (Lafferty et al., 2001b) and thin junction trees (Bach and Jordan, 2002).

In this chapter we have proposed novel algorithms for learning *fully observed* undirected graphical models. When the models are *partially observed*, a standard method to train them is the EM algorithm. When the posterior distribution is intractable, a number of researchers have looked at approximating the E steps with loopy belief propagation (Frey and Kannan, 2001).

Because there is no global cost function which both the E and M steps are minimizing, we cannot make any statements on the accuracy or convergence properties of such algorithms. An exciting research direction is to extend our framework to the partially observed case, where we now have an approximate EM algorithm where both E and M steps are derived as fixed point equations minimizing a region-based free energy.

As a final remark, there have been many approximate free energy methods proposed for approximate inference. Many of these methods can be applied to learning partially observed directed graphical models. An example of this are the variational methods. In this chapter we have shown that these methods can also be applied to learning undirected graphical models. To do this, we looked at maximum likelihood learning as solving a maximum entropy problem, and applied the approximations to the entropy instead. This insight opens up many possibilities for approximate learning in undirected graphical models, some of which have been described in detail here.

Acknowledgments and Remarks

An earlier version of chapter first appeared in the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS) (Teh and Welling, 2003). We thank Kevin Murphy for questioning whether the theory of chapter 5 can be applied to learning, and Francis Bach and Michael Jordan for interesting discussions. We thank Geoffrey Hinton for inspiration and support for this work, and the Ontario government for funding this work.

Chapter 7

Discussion

In this thesis we described contributions to modelling with PoEs and EBMs, and to performing approximate inference and learning using Bethe free energies.

7.1 Products of Experts and Energy-based Models

PoEs are a new class of flexible density models for unsupervised learning. They are obtained by combining simpler density models (experts) by multiplying their densities together and normalizing. The advantage of such a combination is that data is represented efficiently in a distributed fashion across the experts, while keeping inference tractable. To deal with the intractable partition function during learning, contrastive divergence learning is employed.

Initial applications of PoEs to continuous-valued domains were not very successful. In chapters 3 and 4 we described new classes of PoEs that are effective for modelling continuous-valued data.

First we extended RBMs so that they can model discretized continuous-values. This is done by replicating the units in the RBM and using each group of replicas to model a single discretized value. We showed that this RBMrate model can be successfully applied to the tasks of face modelling and recognition.

For face recognition, we used an interesting model where pairs of faces belonging to the

same individual are modelled, rather than single faces. This allows us to model well both the features common to face images of the same individual (e.g. moustache, shape of nose), *and* the facial changes which do not affect the identity of the individual (e.g. mouth movements, glasses). To identify the individual in a test image, we have to pair the test image with every gallery image and find the gallery image most well matched to the test image. This requires a model in which inference can be done easily and cheaply – a role perfect for a PoE. Another class of models where inference can be done easily and cheaply is the linear Gaussian models, e.g. principal components analysis or factor analysis. It will be interesting to compare using these simple models, which have proved quite successful for face recognition, to using PoEs to model pairs of faces. However, we should note that PoEs can model much more complex distributions. This allows more room for improvements which will hopefully translate to better face recognition performance.

Another interesting direction of research is to determine which experts in a PoE contribute most to improving face recognition rates, and to use only those experts. This can further improve the efficiency of inference, and even improve recognition rates, since experts which have not captured useful features of faces and are only contributing noise to the similarity measure, will be pruned. This pruning of experts can be done using cross validation, or class discriminability analysis Bartlett et al. (2002).

RBMrate is not entirely satisfactory as a model for continuous-valued domains, since the data has to be bounded and discretized. EBMs are much more natural models for continuous-valued data. EBMs are generalizations of PoEs in that each expert now need not be a normalized distribution – each expert (called an energy term here) only has to be some easily computed function of the data. The only constraint is that the integral over the data space of $e^{-E(x)}$ where $E(x)$ is the sum of energy terms must not be infinite. This constraint is needed so that the EBM actually defines a proper distribution over the data space. This is typically easy to impose by restricting the functional form of the experts (e.g. letting them be the negative log of heavy-tailed density functions).

Although in chapter 4 we have only explored EBMs with a single layer of linear features,

EBMs can be very flexible models of data. For example, EBMs can be defined using feed-forward or recurrent neural networks (e.g. Welling et al., 2003). They can also be defined for dynamical data using recurrent neural networks. In future work, we would like to explore the various extensions to multiple layers, recurrent connections, and dynamical systems. Another obvious future direction is to apply EBMs to face recognition.

7.2 Bethe Free Energy Approximations

After Yedidia et al. (2001) showed that the fixed points of loopy BP are exactly the stationary points of the Bethe free energy, the next important steps are to understand what is the Bethe free energy, and to understand the relationship between the *stable* fixed points of loopy BP and the Bethe free energy.

The initial aim of our work here is to derive an algorithm to directly minimize the Bethe free energy. In Welling and Teh (2001, 2003) we derived such an algorithm for Boltzmann machines based on gradient descent. This led to a better understand of the nature of the Bethe free energy in terms of Plefka's expansion. We also showed that the naive mean field and naive TAP methods are low order truncations of the Bethe free energy. In experiments we verified that the mean field and TAP approximations are less accurate than the Bethe free energy approximation.

In chapter 5 we derive another algorithm to directly minimize the Bethe free energy. This algorithm works for the more general case of discrete pairwise Markov networks. The algorithm is based on two ideas. Firstly, it uses the fact that the Bethe free energy is convex when the network is tree structured. Secondly, it makes use of an interesting relationship between loopy BP and IS – that they are both fixed point equations derived from trying to minimize the Bethe free energy subject to constraints on certain node marginals.

The algorithm we derived, called UPS, is one of the first algorithms that directly minimizes the Bethe free energy. Another such algorithm is CCCP by Yuille (2002). In contrast with UPS, which is based on the graph theoretic intuition that the Bethe free energy is convex if the

graph is a tree, CCCP is based on the algebraic intuition that the Bethe free energy consists of convex and concave terms. Like UPS, CCCP is a double loop algorithm. At each iteration of the outer loop, the concave terms are upper bounded linearly. The resulting bound on the Bethe free energy is then convex and fixed point equations are derived that are guaranteed to find the global minimum. The inner loop then consists of running these fixed point equations until convergence. This convex-concave decomposition is very powerful, as is seen in that fact that it is easily adapted to the more general case of the Kikuchi free energy. In an exciting recent development, Heskes (2003) uses a similar convex-concave decomposition to show that the stable fixed points of loopy BP are minima of the Bethe free energy.

The relationship between loopy BP and IS, along with the fact that IS is a standard algorithm to learn the parameters of undirected graphical models, allows us to apply the ideas developed in chapter 5 to learning. This is described in chapter 6. In particular, we first describe UPS-JT, a more efficient version of the standard effective IPF procedure of Jiroušek and Přeučil (1995). Then using the fact that the Bethe free energy is the simplest of a hierarchy of approximate free energies based on region graphs (Yedidia et al., 2002), we then proposed a range of approximate learning algorithms based on free energies from this hierarchy. We showed that we can get good results on a simple experiment on hand-written digits using junction graphs.

Unfortunately, because the Bethe free energy and its generalizations based on region graphs are not in general convex, loopy IS is not guaranteed to converge. In the future it would be interesting to develop convergent alternatives to loopy IS. Another interesting direction, taken by Wainwright et al. (2002a,b), is to start with a convexified form of the Bethe free energy that lower bounds the log likelihood. Because the cost function is convex, the derived fixed point equations (called tree-reweighted BP) are guaranteed to find the global optimum. In the future it will be interesting to compare the accuracy of tree-reweighted BP versus loopy IS and to analyze their accuracies c.f. Wainwright et al. (2002b). Another interesting direction of research is to derive convexified region graph free energies that give tighter bounds on the log likelihood.

Bibliography

S.A. Abdallah and M.D. Plumbley. If edges are the independent components of natural images, what are the independent components of natural sounds? In *International Conference On Independent Component Analysis and Blind Signal Separation*, 2001.

S.M. Aji and R.J. McEliece. The generalized distributive law and free energy minimization. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, volume 39, 2001.

S. Amari, A. Cichocki, and H. Yang. A new algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, volume 8, pages 757–763, 1996.

S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematica Monographs*. Oxford University Press, 2000.

H. Attias. Independent factor analysis. *Neural Computation*, 11:803–851, 1999.

F.R. Bach and M.I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, volume 14, 2002.

M.S. Bartlett, J.R. Movellan, and T.J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 2002. In Press.

M.J. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In *Bayesian Statistics*, volume 7, 2003.

A.J. Bell and T.J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

P.N. Belmumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces versus fisherfaces: recognition using class specific linear projection. In *European Conference on Computer Vision*, 1996.

A. Berger. The improved iterative scaling algorithm: A gentle introduction. `http://www.cs.cmu.edu/~aberger/maxent.html`, 1997.

A. Berger, S. Della Pietra, and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), 1996.

J. Besag. Discussion of paper by P. Switzer. *Bulletin of The International Statistical Institute*, 50:422–425, 1983.

A. Brown and G.E. Hinton. Products of hidden Markov models. In *Proceedings of Artificial Intelligence and Statistics*, 2001.

J.F. Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters*, 4(4):112–114, 1997.

S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

P. Clifford. Markov random fields in statistics. In G.R. Grimmerr and D.J.A. Welsh, editors, *Disorder in Physical Systems*. Oxford Science Publications, 1990.

P. Comon. Independent component analysis: A new concept? *Signal Processing*, 36:287–314, 1994.

G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

R. Cowell. Introduction to inference for Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.

P. Dayan, G.E. Hinton, R.M. Neal, and R.S. Zemel. Helmholtz machines. *Neural Computation*, 7:1022–1037, 1995.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

W.E. Deming and F.F. Stephan. On a least square adjustment of a sampled frequency table when the expected marginal totals are known. *Annals of Mathematical Statistics*, 11:427–444, 1940.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

W. Freeman and E. Pasztor. Learning to estimate scenes from images. In *Advances in Neural Information Processing Systems*, volume 11, 1999.

B. Frey and A. Kannan. Accumulator networks: suitors of local probability propagation. In *Advances in Neural Information Processing Systems*, volume 13, 2001.

B. Frey and D.J.C. MacKay. A revolution: Belief propagation in graphs with cycles. In *Neural Information Processing Systems*, volume 10, 1997.

B.J. Frey, R. Patrascu, T. Jaakkola, and J. Moran. Sequentially fitting "inclusive" trees for inference in noisy-OR networks. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

N. Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 14, 1998.

A. Georges and J.S. Yedidia. How to expand around mean-field theory using high-temperature expansions. *Journal of Physics A*, 24:2173–2192, 1991.

C.J. Geyer. Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics: Proceedings of the Symposium on the Interface*, 23:156–163, 1991.

Z. Ghahramani and M.J. Beal. Variational inference for Bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, volume 12. The MIT Press, 2000.

Z. Ghahramani and M.J. Beal. Propagation algorithms for variational Bayesian learning. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

M. Girolami. A variational method for learning overcomplete representations. *Neural Computation*, 13:2517–2532, 2001.

S.S. Gupta, editor. *Differential Geometry in Statistical Inference*, volume 10 of *Lecture Notes – Monograph Series*. Institute of Mathematical Statistics, 1987.

D. Heckerman. A tutorial on learning Bayesian networks. In M.I. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1998.

T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

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

G.E. Hinton, B. Sallans, and Z. Ghahramani. A hierarchical community of experts. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

G.E. Hinton and T.J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press, 1986.

G.E. Hinton and Y.W. Teh. Discovering multiple constraints that are frequently approximately satisfied. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 227–234, 2001.

G.E. Hinton, Y.W. Teh, M. Welling, and S. Osindero. Contrastive backpropagation. In preparation, 2002.

G.E. Hinton, M. Welling, Y.W. Teh, and S. Osindero. A new view of ICA. In *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation*, volume 3, 2001.

A. Hyvärinen and P.O. Hoyer. A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Research*, 41(18):2413–2423, 2001.

A. Hyvärinen and M. Inki. Estimating overcomplete independent component bases for image windows. *Journal of Mathematical Imaging and Vision*, 2002. in press.

F.V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

R. Jiroušek and S. Přeučil. On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics And Data Analysis*, 19:177–189, 1995.

M.I. Jordan, editor. *Learning in Graphical Models*. Kluwer Academic Pubilshers, 1998.

M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

H. Kappen and P. Rodriguez. Mean field theory for graphical models. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods*. The MIT Press, 2001.

H.J. Kappen and F.B. Rodríguez. Efficient learning in Boltzmann machines using linear response theory. *Neural Computation*, 10:1137–1156, 1998.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, volume 18, 2001a.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001b.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

S.L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50, 1988.

D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, October 1999.

J. Lee. New Monte Carlo algorithm : Entropic sampling. *Physical Review Letters*, 71(2): 211–214, 1993.

M.S. Lewicki and B.A. Olshausen. A probabilistic framework for the adaptation and comparison of image codes. *J. Opt. Soc. Am. A: Optics, Image Science, and Vision*, 16(7): 1587–1601, 1999.

M.S. Lewicki and T.J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12:337–365, 2000.

D.J.C. MacKay. Maximum likelihood and covariant algorithms for independent components analysis. http://wol.ra.phy.cam.ac.uk/mackay/abstracts/ica.html, 1996.

D.J.C. MacKay. Failures of the one-step learning algorithm. http://wol.ra.phy.cam.ac.uk/mackay/abstracts/gbm.html, 2001.

D.J.C. MacKay and R.M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32:1645–1646, 1996.

J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.

S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions in Signal Processing*, 41(12):3397–3414, 1993.

E. Marinari and G. Parisi. Simulated tempering : A new Monte Carlo scheme. *Europhysics Letters*, 19:451–458, 1992.

G. Mayraz and G.E. Hinton. Recognizing hand-written digits using hierarchical products of experts. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

R. McEliece, D.J.C. MacKay, and J. Cheng. Turbo decoding as an instance of pearl's belief propagation algorithm. *IEEE J. Selected Areas in Communication, 1997*, 16:140–152, 1998.

T.P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2001.

B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

B. Moghaddam, W. Wahid, and A. Pentland. Beyond eigenfaces: probabilistic matching for face recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.

Q. Morris. *Recognition Networks for Approximate Inference in BN2O Networks*. PhD thesis, Massachessets Institute of Technology, 2002.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference : An empirical study. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 15. Morgan Kaufmann Publishers, 1999.

R.M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Department of Computer Science, 1993.

R.M. Neal. Sampling from multimodel distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1996.

R.M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.

R.M. Neal. Circularly-coupled Markov chain sampling. Technical Report 9910 (revised), University of Toronto, Department of Statistics, 2002.

R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workship on Machine Learning for Information Filtering*, 1999.

B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1. *Vision Research*, 37:3311–3325, 1997.

B.A. Olshausen and K.J. Millman. Learning sparse codes with a mixture-of-gaussians prior. In *Advances in Neural Information Processing Systems*, volume 12, pages 841–847, 2000.

M. Opper and O. Winther. From naive mean field theory to the TAP equations. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods : Theory and Practice*. The MIT Press, 2001.

J. Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo CA, 1988.

B.A. Pearlmutter and L.C. Parra. A context-sensitive generalization of ICA. In *Proceedings of the International Conference on Neural Information Processing*, 1996.

P.J. Phillips, H. Moon, P. Rauss, and S.A. Rizvi. The FERET september 1996 database and evaluation procedure. In *International Conference on Audio and Video-based Biometric Person Authentication*, 1997.

T. Plefka. Convergence condition of the TAP equations for the infinite-ranged Ising sping glass model. *Journal of Physics A*, 15:1971, 1982.

J.G. Propp and D.B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252, 1996.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing : Explorations in The Microstructure of Cognition. Volume 1 : Foundations*. The MIT Press, 1986.

D. Saad and M. Opper, editors. *Advanced Mean Field Methods : Theory and Practice*. The MIT Press, 2001.

B. Sallans. A hierarchical community of experts. Master's thesis, University of Toronto, Canada, 1998.

B. Sallans and G.E. Hinton. Using free energies to represent Q-values in a multiagent reinforcement learning task. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

G.R. Shafer and P.P. Shenoy. Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–352, 1990.

O. Shriki, H. Sompolinsky, and D.D. Lee. An information maximization approach to overcomplete and recurrent representations. In *Advances in Neural Information Processing Systems*, volume 14, pages 612–618, 2002.

E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multi-scale transforms. *IEEE Transactions Information Theory*, 38(2):587–607, 1992.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press, 1986.

T. Tanaka. Information geometry of mean field approximation. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods*. The MIT Press, 2001.

S. Tatikonda and M.I. Jordan. Loopy belief propagation and Gibbs measures. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 18, 2002.

Y.W. Teh and G.E. Hinton. Rate-coded restricted Boltzmann machines for face recognition. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

Y.W. Teh and M. Welling. Passing and bouncing messages for generalized inference. Technical Report 001, Gatsby Computational Neuroscience Unit, UCL, 2001.

Y.W. Teh and M. Welling. The unified propagation and scaling algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2002.

Y.W. Teh and M. Welling. On improving the efficiency of the iterative proportional fitting procedure. In *Proceedings of Artificial Intelligence and Statistics*, 2003.

Y.W. Teh, M. Welling, S. Osindero, and G.E. Hinton. Energy-based models for sparse overcomplete representations. Submitted to Journal of Machine Learning Research, 2003.

D. J. Thouless, P. W. Anderson, and R. G. Palmer. Solution of a 'Solvable model of a spin glass'. *Phil. Mag.*, 35:593, 1977.

M.E. Tipping and C.M. Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, 1997.

M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3 (1):71–86, 1991.

J.H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London B*, 265:359–366, 1998.

M.J. Wainwright. *Stochastic Processes on Graphs with Cycles: Geometric and Variational Approaches*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2002.

M.J. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 18, 2002a.

M.J. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *Advances in Neural Information Processing Systems*, volume 14, 2002b.

Y. Weiss. Belief propagation and revision in networks with loops. Technical Report A.I. Memo 1616 and C.B.C.L. Paper 115, Massachusetts Institute of Technology, 1997.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitraty graphs. *IEEE Transactions on Information Theory*, 47(2): 736–744, 2001.

M. Welling, G.E. Hinton, and S. Osindero. Learning sparse topographic representations with products of student-t distributions. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

M. Welling and Y.W. Teh. Belief optimization for binary networks: a stable alternative to loopy belief propagation. In *Uncertainty in Artificial Intelligence*, 2001.

M. Welling and Y.W. Teh. Approximate inference in boltzmann machines. To appear in Artificial Intelligence, 2003.

P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

C.K.I. Williams and F.V. Agakov. An analysis of contrastive divergence learning in Gaussian Boltzmann machines. Technical Report EDI-INF-RR-0120, Institute for Adaptive and Neural Computation, Division of Informatics, University of Edinburgh, 2002.

J.S. Yedidia. An idiosyncratic journey beyond mean field theory. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods*. The MIT Press, 2001.

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, volume 13, pages 689–695, 2001.

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2002-35, Mitsubishi Electric Research Laboratories, 2002.

A.L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.

S.C. Zhu, Y.N. Wu, and D. Mumford. Minimax entropy principle and its application to texture modelling. *Neural Computation*, 9(8), 1997.