The Cortex as a Graphical Model

An Investigation of Learning Rules in Undirected Graphical Models and an Initial Exploration of Their Relationship to Known Properties of Cortical Anatomy and Activity

Jason Tyler Rolfe
Computation and Neural Systems
California Institute of Technology

August 14, 2006

1 Introduction

The world which we perceive is not the world of the senses. Our eyes detect lines and splotches of color; our ears hear a cacophony of ever-changing tones; our skin detects gradients of pressure and heat, yet we experience a world of unitary objects, continuous in time and space. One of the primary functions of the brain is to infer the state of the world from the incomplete and uncertain information provided by the senses. It is only by combining multiple cues and correlating them with past experience that the brain is able to extract order from the chaos of sight, sound, touch, smell, and taste.

The insufficiency of current sensory input is especially apparent in the visual system. For instance, the fovea of the eye perceives the world at considerably higher resolution and with richer color than the periphery of the eye, yet we rarely notice that the sensitivity of our visual field is not uniform [15]. Similarly, figure-ground discrimination is often ambiguous. The differences in luminosity within an object due to inhomogeneities in lighting conditions, texture, and color, are frequently much greater than those between an object and its surrounding environment [15]. Only a fraction of the contours we see actually correspond to the borders of objects, but we have no difficulty isolating objects from their visual environment. Object identification suffers from ambiguities that are even more severe. Humans can recognize objects from many angles and despite substantial occlusion. Indeed, novel instances of familiar classes of objects can be identified, even though they do not share all of the attributes of any previously encountered example.

In all of these cases, many individual pieces of information must be combined to yield our final perception. To construct the perceived visual field beyond the fovea, previous visual input is combined with current stimuli to yield a unified whole. Figure-ground discriminations are based upon the patterns of luminance and color throughout the object and its background. Objects are identified on the basis of numerous separate features. For instance, eyes, ears, a nose, and a mouth in the appropriate configuration suggest the presence of a face.

In accordance with the structure of visual perception tasks, the computations necessary to perform these tasks require the global synthesis of numerous local computations. The curves which comprise the boundaries of visual objects are by definition continuous, so the line which separates an object from its background can be identified by locally moving from point to point along its border. These local computations must then be brought together to pick out the figure as a whole. Similarly, the individual visual features by which objects are identified can be processed separately, but must be judged as an aggregate to perform accurate object recognition.

The distributed nature of visual processing is also reflected in the organization of areas of the brain associated with visual information. In the early visual system, the location of a neuron corresponds to the position of its receptive field [15]. Neurons in V1 are organized into columns with consistent occular dominance, and smaller sub-columns which respond to the same line orientation. Connections between columns are ordered, and preferentially target neurons with similar receptive fields [49]. Indeed, throughout the brain, sensory processing is performed by a topographically structured network [34]. In the auditory system, the receptive fields are topographically mapped according to preferred frequency in regions as high as the primary auditory cortex. Primary sensory cortex contains a topographic mapping of the surface of the body [34].

Thus, both the computational demands faced by the brain and its underlying anatomy suggest the importance of networks of small computational units with local connectivity. Numerous mathematical models have been proposed to elucidate the operation of primarily local networks of neurons in the brain [7, 12]. Ideally, these models should

accurately capture the dynamics of real neurons and the topology of their connections, while remaining analytically tractable. Neurons, however, are composed of multiple regions which interact nonlinearly. While detailed biophysical models have been constructed to describe the electrical and chemical dynamics of neurons, they can only be solved numerically, depend upon many parameters which are difficult to estimate accurately, and require massive numbers of CPU cycles to simulate even a single neuron [17].

Worse still, neurons are connected into hierarchical networks in which recurrent connections occur at every level. The difficulty of analyzing and designing feedback systems is apparent from the prevalence of primarily feedforward designs in human constructions ranging from combinational circuitry to artificial neural networks. When feedback is incorporated into human-designed systems, such as in recursive computer programs or analog circuitry, it is used with great care. Positive feedback loops are as easily created as negative feedback loops, but their instability tends to destroy potentially essential data. Feedback can also give rise to chaotic dynamics, which exhibit sensitive dependence upon initial conditions and so are inappropriate for many types of computation in which very small changes in the input should not substantially affect the output.

The brain is awash in feedback, and most long-range connections are excitatory [8]. Many prominent neural pathways which are commonly thought of as transmitting information from "lower" to "higher" areas, such as the projection from the LGN to V1 in the visual stream, are composed primarily of feedback projections. Lateral connections within a given area, such as between the cortical columns in V1, introduce further feedback cycles [3]. When uncontrolled, as in epileptic seizures, feedback can also be devastating to the brain, but its ubiquitous presence in neural circuits suggests that it also serves as the basis for cognitive function.

The two most prominent models of recurrent neural connections are Hopfield networks and their generalization, Boltzmann machines, which will be discussed rigorously in section 4 [12]. Hopfield networks were initially formulated as a model of memory storage and recall [13]. Amongst their notable features, Hopfield networks can reconstruct previously stored memories from corrupted inputs. However, since every unit directly represents one of the bits of the stored activity patterns, Hopfield networks cannot represent more than second-order statistics of the input data ($\langle X_i X_j \rangle$ for all variables X_i and X_j). Second-order statistics often carry little useful information. The set of all n bit numbers with parity 0 ($n > 2$) has $\langle X_i X_j \rangle = 0.25$ for all X_i and X_j. This property also holds for the set of all n bit numbers with parity 1, as well as the set of four elements where one element has all 1's, whereas the other three have only one 1. One can easily imagine that other meaningful data sets, such as a set of black-and-white pictures of an object from multiple angles, would not be accurately captured by second-order statistics.

By introducing hidden units the state of which is not defined by the stored activity patterns, Boltzmann machines allow the representation of a much larger set of patterns. In addition to remembering past input patterns, Hopfield networks and Boltzmann machines can also perform statistical inference. If some of the inputs are fixed and others are allowed to change freely, the free variables will assume states according to the statistics of the input patterns [39].

The computational units of Hopfield networks and Boltzmann machines are generally considered to be analogous to single neurons, and the interactions of these units are intended to capture the activity of small networks of neurons [12]. However, the utility of these constructions as models of actual neural dynamics is limited by the dissimilarities between their fundamental elements and actual neurons. In particular, neural synapses are unidirectional, whereas Hopfield networks and Boltzmann machines assume symmetric bidirectional connections. Moreover, the complexity of actual neural dynamics is necessarily ignored by these simple models.

Hopfield networks and Boltzmann machines are both instances of a larger class of probabilistic systems called undirected graphical models [18]. An undirected graphical model defines a probability distribution that can be factored into multiple potential functions (or relations), each of which depends on only a subset of the available variables. Multiplicative factorization is a reasonable way to break a large function of many variables into smaller functions, although by no means the only way. For instance, the function could be decomposed additively or recursively. The brain certainly uses a decomposition in which each of the component units is not a function of all possible inputs, since each region of the brain receives only a fraction of all possible inputs. Graphical models are particularly interesting since, using the belief propagation algorithm, it is easy and efficient to calculate the marginal probability of one variable given the other variables over their probability distribution. This operation is also known as Bayesian inference.

Bayesian inference accurately captures many features of the operation of the visual system, such as incorporation of prior probabilities, cue integration, and perceptual explaining away [19, 20]. Since Bayesian inference can be easily performed within graphical models using the belief propagation algorithm, this suggests that the visual system may implement a graphical model. Graphical models, both directed and undirected, have previously been proposed as direct models of brain function [9, 25, 43]. However, these models have generally been constructed at what Marr described as the computational level [30]. That is, they attempted to capture the sort of operations performed by the brain, but not how those operations are executed. In contrast, in Section 5 we will offer an algorithmic and

implementational model of cortical function based on the belief propagation algorithm used to perform statistical inference in undirected graphical models. We will propose an input-output function computed by individual neurons and suggest possible biophysical mechanisms which might underly this operation. This algorithmic model of individual neurons will lead to a computational model of the cortex at the level of cortical columns.

Both anatomical and functional evidence indicates that the cortex of the brain is organized into columns, between 300 and 600 μm in diameter, which project perpendicularly to the surface of the brain through all six layers of cortex [34, 40]. Cells in a single column respond to similar stimuli, whereas cells in adjacent columns respond to different stimuli. Individual columns are strongly recurrently connected, and long-distance projections out of the column are organized in patches the size of which corresponds to that of a single column [3]. It is thus reasonable to conjecture that the cortical column functions as a single computational unit with a coherent state. The apparently discrete connections between pairs of columns is topologically similar to the edges and nodes of an undirected graphical model [3, 49]. With this as our inspiration, we will find in Section 5 that there are considerable additional similarities between networks of cortical columns and undirected graphical models.

In addition to its impressive computational abilities, the brain is also able to learn from experience. Any model of cortical dynamics that does not account for this critical feature is necessarily incomplete. In Sections 2 and 3 we present some new developments in techniques for training undirected graphical models, and show in Section 5 how they can be implemented directly in our model of cortex as an undirected graphical model. Since Boltzmann machines are a restricted class of undirected graphical models, our algorithms can also be applied in that context, as discussed in Section 4.

1.1 Undirected Graphical Models

Undirected graphical models can be defined in terms of a factor-graph representation (see Figure 1) [23]. Consider a set of variables, of which some ($Y = \{Y_1, Y_2, \ldots\}$) are directly associated with observable quantities in the external world and some ($H = \{H_1, H_2, \ldots\}$) are internal (often called "hidden") quantities. Each random variable takes on one of many possible values, although we will assume that each variable can only assume a finite set of values. The set of all variables is defined as the Cartesian product $X = Y \times H = \{Y_1, Y_2, \ldots Y_m, H_1, H_2, \ldots, H_n\}$, so that each X specifies a value for every Y_i and every H_i.

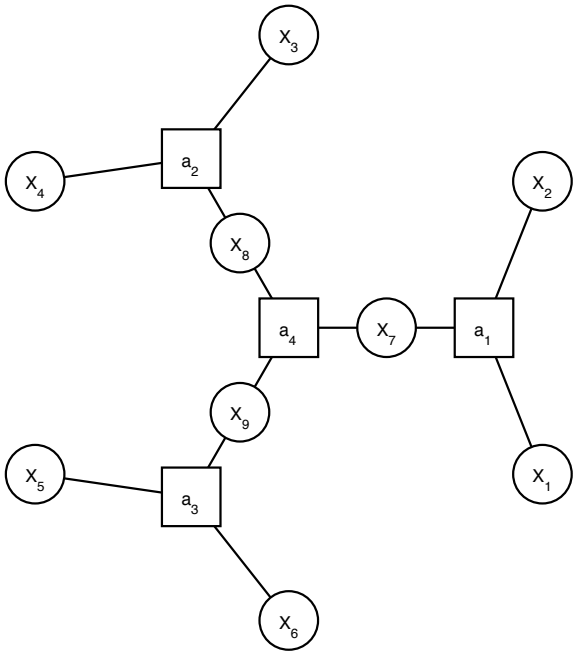


Figure 1: A factor graph

The variables of the undirected graphical model are connected together by a set of relations (or factors, and hence "factor graph"). This can be visualized as a bipartite graph (see Figure 2), where both relations and variables correspond to nodes. Edges exist between a subset of the relation node-variable node pairs; relations are never connected to other relations, and variables are never connected to other variables. Each relation node a is associated with a non-negative function f_a, which takes as arguments the variables to which node a is connected in the factor graph. We will interchangeably refer to a and f_a as "the relation." Since the arguments of f_a are the subset of $\{Y_i\} \cup \{H_i\}$ connected to a in the factor graph, we write $f_a(X_a)$ to indicate that f_a depends on a subset of X particular to a. We often refer to a particular assignment of values to the variables connected to relation node a as a configuration of X_a.

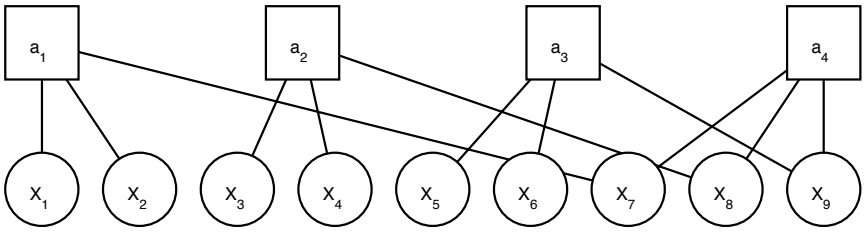


Figure 2: A bipartite graph

The probability distribution associated with an undirected graphical model is defined by

$$p(X) = \frac{1}{Z} \prod_a f_a(X_a)$$

where Z is a normalizing factor called the "partition function," defined as

$$Z = \sum_X \left(\prod_a f_a(X_a) \right)$$

In addition to the partition function Z, we shall also sometimes refer to the "norm" of relation a' and local configuration $X_{a'} = x$. This term is defined to be

$$\text{Norm}_{a',x} = \sum_{X | X_{a'} = x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)$$

Note that $\sum_{X_{a'}} f_{a'}(X_{a'}) \text{Norm}_{a',X_{a'}} = Z$.

Intuitively, the relations of an undirected graphical model manifest the degree to which their arguments are in a high-probability configuration. If the variables are chosen such that the values of all the relations are large (relative to their average values), then that assignment of values to variables has a high probability. Alternatively, the relations can be thought of as indicating the e^{-H}, where H is the energy of the local configuration. Multiplying the relationships together corresponds to adding their contributions to the total energy. Large values of the relation are equivalent to small energies, which are most likely under the Boltzmann distribution. This interpretation will prove especially apt in the case of Boltzmann machines, discussed in Section 4.

Graphical models can be trained to represent a desired probability distribution by altering the function associated with each relation. Each variable can only take on a discrete set of values, so a function f_a corresponds to a finite set of parameters. When we write $f_a(X_a = x)$, we mean the value of some particular relation a when its arguments assume the specific configuration $X_a = x$. Since $f_a(X_a = x)$ is a real number, it is possible to take derivatives with respect to it. We often refer to the collection of all such values for all the relations as the parameters of the graphical model, or θ. Thus, $p(X|\theta)$ is the probability distribution of a graphical model given a particular choice of all the f_a.

We indicate a set of n assignments of values to the variables $\{Y_i\} \cup \{H_i\}$ by $\mathbf{X} = \left\{X^1, X^2, \ldots, X^n\right\}$. Correspondingly, a set of assignments of values to the observed variables is denoted by $\mathbf{Y} = \left\{Y^1, Y^2, \ldots\right\}$ and a set of assignments of values to the hidden variables is denoted by $\mathbf{H} = \left\{H^1, H^2, \ldots\right\}$. As stated above, we sometimes refer

to an assignment of values to a set of variables as a configuration of the variables. We implicitly assume throughout this thesis that input configurations (i.e., the assignments of values to the visible variables) are independent from trial to trial, so $p(\mathbf{Y}) = p(Y^1) \cdot p(Y^2) \cdots$ The conditional probability of a configuration X of the variables given a particular setting θ of the parameters (i.e., given a particular choice of the values of the relations $f_a(X_a)$ for all possible settings of their arguments) is written as $p(X|\theta)$. To indicate the subset of a variable configuration X corresponding to the visible variables, we write $Y(X)$. Thus, $Y(X) = Y^i$ indicates that the visible variables of X match the input configuration Y^i, and $X|Y(X) = Y^i$ indicates the set of all configurations of the variables such that the visible variables match the input configuration Y^i.

1.2 Maximizing Likelihood

One of the primary functions of the brain is to model the outside world. Every human task, from conceptual planning to simple movement requires a model of our body and environment. Consider the possibility that cortical columns implement an undirected graphical model. One reasonable way to set the probability distribution of such an undirected graphical model is to attempt to capture the probability distribution of the visible variables seen in the outside world. In the case of the visual system, images of commonly seen objects would have a high probability, whereas images containing completely novel objects or noisy images which do not contain separate objects would have a low probability. Such a probability distribution would have a variety of useful features. For instance, it could be used to infer hidden portions of an image: the unseen component with the largest marginal probability conditioned on the observed portion of the picture is the most likely completion of the image. The application of a Gibbs sampling procedure, updating each variable according to its marginal probabilities in the graphical model, could remove noise from corrupted images. This technique is equivalent to that used to recall memories in a Hopfield network [12], and its implementation in an appropriately-structured undirected graphical model is similar to belief propagation [54]. The hidden variables in the graphical model may capture important abstract features of the inputs. Unlike principal component analysis or singular value decomposition, these hidden variables are best thought of as classifying the input into one of a finite number of categories, rather than projecting it into a low-dimensional space. Thus, object recognition and categorization may be possible on the basis of the hidden variables without the use of an additional clustering algorithm

Human experience is finite, so we can never know the actual probability of every possible set of observations. Instead, we are presented with a sample of all possible observations, where the frequency of each observation is proportional to its probability. Given the values of the observed variables $\mathbf{Y} = \left\{Y^i\right\}$ in observations $i = 1, \ldots, n$, we can build a model, θ, of this data by maximizing the likelihood of the data given the relations, $p(\mathbf{Y}|\theta)$. The selection of the likelihood as the quantity to be maximized is a philosophical decision. We could just as well try to maximize the posterior probability $p(\theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\theta)p(\theta)}{p(\mathbf{Y})} = \frac{p(\mathbf{Y}|\theta)p(\theta)}{\sum_\theta p(\mathbf{Y}|\theta)p(\theta)}$ Of course, this more Bayesian approach requires a choice of priors for θ. The posterior probability with a uniform prior is equivalent to the likelihood. We have no particular knowledge regarding the prior probability of various choices of θ, so we will focus on the likelihood. Since the logarithm is a monotonic function, maximizing the likelihood is equivalent to maximizing the log likelihood. We will find this latter quantity easier to work with, since

$$\log p(\mathbf{Y}|\theta) = \log \left(\prod_i p(Y^i|\theta) \right) = \sum_i \log p(Y^i|\theta)$$

We will need to take the derivative to maximize the likelihood, and the derivative of a sum is equal to the sum of the derivatives, whereas the derivative of a large product contains many high-order terms.

The probability distribution of a graphical model is defined in terms of all its variables, not just the observed variables. To calculate the marginal likelihood of the observed data alone, we must use $p(Y|\theta) = \sum_{X|Y(X)=Y} p(X|\theta)$. The marginalization implicit in the probability of each observed data point will prevent the log from passing through to individual probabilities in the calculation of the log likelihood:

$$\log p(\mathbf{Y}|\theta) = \sum_i \log p(Y^i|\theta) = \sum_i \log \sum_{X|Y(X)=Y^i} p(X|\theta)$$

2 Gradient-based Maximization

The most well-known algorithm for optimizing the likelihood of the data is simple gradient ascent on the parameters [36]. In the Boltzmann machine, this gives rise to the wake-sleep algorithm [12].

$$
\begin{aligned}
\frac{\partial \log p\left(\mathbf{Y}|\theta\right)}{\partial f_{a'}(X_{a'}=x)} &= \frac{\partial}{\partial f_{a'}(X_{a'}=x)} \log \prod_i p\left(Y^i|\theta\right) \\
&= \sum_i \frac{\partial}{\partial f_{a'}(X_{a'}=x)} \log p\left(Y^i|\theta\right) \\
&= \sum_i \frac{1}{p(Y^i|\theta)} \cdot \frac{\partial}{\partial f_{a'}(X_{a'}=x)} \left(p\left(Y^i|\theta\right)\right) \\
&= \sum_i \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} \cdot \frac{\partial}{\partial f_{a'}(X_{a'}=x)} \left(\frac{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)}{\sum_X \prod_a f_a(X_a)} \right) \\
&= \sum_i \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} \cdot \left(\frac{\left(\sum_{X|Y(X)=Y^i, X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)\right) \cdot \left(\sum_X \prod_a f_a(X_a)\right)}{\left(\sum_X \prod_a f_a(X_a)\right)^2} - \right. \\
&\qquad \left. \frac{\left(\sum_{X|X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)\right) \cdot \left(\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)\right)}{\left(\sum_X \prod_a f_a(X_a)\right)^2} \right) \\
&= \sum_i \left(\frac{\sum_{X|Y(X)=Y^i, X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} - \frac{\sum_{X|X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_X \prod_a f_a(X_a)} \right) \\
&= \frac{1}{f_{a'}(X_{a'}=x)} \sum_i \left(p(X_{a'}=x|Y(X)=Y^i) - p(X_{a'}=x)\right)
\end{aligned}
$$

In the wake-sleep and related algorithms, each $f_{\alpha'}(X_{\alpha'}=x)$ is incremented in proportion to this gradient. The first term within the summation corresponds to the wake stage, where the parameters are changed to reflect the probabilities of the variables in the presence of the observed data, whereas the second term corresponds to the sleep stage, where the parameters are adjusted to forget the old probability distribution over the variables. To actually perform gradient updates, we move by a fixed increment in the direction of the gradient. Thus, it is safe to divide the entire right side by n, the number of training samples, to obtain

$$
\frac{\partial}{\partial f_{a'}(X_{a'}=x)} \log p\left(\mathbf{Y}|\theta\right) \propto \frac{1}{f_{\alpha'}(X_{\alpha'}=x)} \cdot \left(\left\langle p(X_{\alpha'}=x|Y(X)=Y^i)\right\rangle_i - p(X_{\alpha'}=x)\right)
$$

where the angle brackets indicate the mean over the observations $i=1,\ldots,n$.

Rather than simply move in the direction of the gradient, we can maximize the log likelihood by setting

$\frac{\partial}{\partial f_{a'}(X_{a'}=x)} \log p\left(\mathbf{Y}|\theta\right) = 0$:

$$
\begin{aligned}
0 &= \sum_i \left(\frac{\sum_{X|Y(X)=Y^i, X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} - \frac{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)}{\sum_X \prod_a f_a(X_a)} \right) \\
n \cdot \frac{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)}{\sum_X \prod_a f_a(X_a)} &= \sum_i \frac{\sum_{X|Y(X)=Y^i, X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} \\
1 &= \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)} \cdot \sum_i \frac{1}{n} \cdot \frac{\sum_{X|Y(X)=Y^i, X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} \\
f_{a'}(X_{a'}=x) &= \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)} \cdot \sum_i \frac{1}{n} \cdot \frac{\sum_{X|Y(X)=Y^i, X_{a'}=x} \left(\prod_a f_a(X_a) \right)}{\sum_{X|Y(X)=Y^i} \prod_a f_a(X_a)} \\
&= \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)} \cdot \sum_i \frac{1}{n} p(X_{a'}=x|Y^i) \\
&= f_{a'}(X_{a'}=x) \cdot \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_a f_a(X_a) \right)} \cdot \sum_i \frac{1}{n} p(X_{a'}=x|Y^i) \\
&= f_{a'}(X_{a'}=x) \cdot \frac{\left\langle p(X_{a'}=x|Y^i) \right\rangle}{p(X_{a'}=x)}
\end{aligned}
$$

If all variables are observed and thus $p(X_{a'}=x|Y^i)$ is known exactly, we can obtain the iterative proportional fitting procedure by turning this equality into an update rule [50]. Note that while such a manipulation is somewhat ad hoc and convergence is by no means assured, if the relations do converge, they will do so at an extremum of the likelihood. We denote the old values of the relations by $f_a^t(X_a)$ and the new value of relation a' at $(X_{a'}=x)$ after maximizing the log likelihood by $f_{a'}^{t+1}(X_{a'}=x)$.

$$
f_{a'}^{t+1}(X_{a'}=x) \longleftarrow f_{a'}^t(X_{a'}=x) \cdot \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_a f_a(X_a) \right)} \cdot \sum_i \frac{1}{n} p(X_{a'}=x|Y^i)
$$

As we shall see in Section 3, a very similar result can be obtained in the presence of hidden variables using expectation maximization, but in that case the probability on the right side is replaced by the expected value calculated with the old parameters. While the fixed points of this update are indeed the maximum of the log likelihood (as is also the case for gradient updates), a more efficient learning rule might be obtained by solving for $f_{\alpha'}(X_{\alpha'}=x)$ exactly. Unfortunately, this is a difficult operation, since $f_{\alpha'}(X_{\alpha'}=x)$ appears repeatedly on the right

side of the equation.

$$
\begin{aligned}
1 &= \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a(X_a)\right)} \cdot \sum_i \frac{1}{n} \cdot \frac{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_{X|Y(X)=Y^i}\prod_a f_a(X_a)} \\
&= \frac{\sum_X \prod_a f_a^t(X_a) + \left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)\cdot \sum_{X|X_{a'}=x}\prod_{a\neq a'} f_a^t(X_a)}{\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)} \cdot \\
&\quad \sum_i \frac{1}{n} \cdot \frac{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)}{\sum_{X|Y(X)=Y^i}\prod_a f_a^t(X_a) + \left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)\cdot \sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)} \\
&= \left(\frac{\sum_X \prod_a f_a^t(X_a)}{\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)} + f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)\cdot \\
&\quad \sum_i \frac{1}{n} \cdot \frac{1}{\frac{\sum_{X|Y(X)=Y^i}\prod_a f_a^t(X_a)}{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)} + f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)} \\
&\approx \left(\frac{\sum_X \prod_a f_a^t(X_a)}{\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)} + \left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)\right)\cdot \\
&\quad \sum_i \frac{1}{n}\left(\frac{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)}{\sum_{X|Y(X)=Y^i}\prod_a f_a^t(X_a)} - \right. \\
&\qquad \left.\left(\frac{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a^t(X_a)\right)}{\sum_{X|Y(X)=Y^i}\prod_a f_a^t(X_a)}\right)^2 \left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)\right)
\end{aligned}
$$

assuming that $\left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)$ is small and using the Taylor expansion. Solving for $\left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x)\right)$, we obtain

$$
\begin{aligned}
X &= \frac{N_1 - N_2 M \pm \sqrt{(N_2 M - N_1)^2 + 4N_2(N_1 M - 1)}}{2N_2} \\
&= \frac{1}{2}\cdot\left(\frac{N_1}{N_2} - M \pm \frac{\sqrt{(N_2 M + N_1)^2 + 4N_2}}{N_2}\right)
\end{aligned}
$$

where

$$
\begin{aligned}
X &= f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^t(X_{a'}=x) \\
M &= \frac{\sum_X \prod_a f_a^t(X_a)}{\sum_{X|X_{a'}=x}\prod_{a\neq a'} f_a(X_a)} = \frac{f_{a'}(X_{a'}=x)}{p(X_{a'}=x)} \\
N_1 &= \frac{1}{n}\sum_i \frac{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_{X|Y(X)=Y^i}\prod_a f_a(X_a)} = \left\langle \frac{p(X_{a'}=x|Y(X)=Y^i)}{f_{a'}(X_{a'}=x)} \right\rangle \\
N_2 &= \frac{1}{n}\sum_i \left(\frac{\sum_{X|Y(X)=Y^i, X_{a'}=x}\left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_{X|Y(X)=Y^i}\prod_a f_a(X_a)}\right)^2 = \left\langle \left(\frac{p(X_{a'}=x|Y(X)=Y^i)}{f_{a'}(X_{a'}=x)}\right)^2 \right\rangle
\end{aligned}
$$

In contrast, the IPF update rule is equivalent to

$$X = f_{a'}(X_{a'}=x)\cdot(N_1 M - 1)$$

The conjugate gradients method is a common and powerful algorithm for multi-dimensional optimization problems, and so seems like a reasonable technique for optimizing the log likelihood of an undirected graphical model

[38]. However, conjugate gradients requires the ability to maximize the function in question along a line in an arbitrary direction. The safest techniques to perform this operation calculate the value of the function at many points along the line and so find ever-tighter bounds around a local maximum. Unfortunately, evaluating the average log likelihood with a new set of parameters over a large set of data points is prohibitively computationally expensive. We could attempt to find a local maximum along an arbitrary line by setting the first derivative equal to zero, but this strategy also proves intractable. Given a direction in the space of parameters defined by a magnitude d and parameters $\theta_d = \left\{f_{a'}(X_{a'} = x) + w_{X_{a'}=x} \cdot d\right\}_{a', X_{a'}=x}$, we find

$$\begin{aligned}
\frac{\partial \log p\left(\mathbf{Y}|\theta_d\right)}{\partial d} &= \frac{\partial}{\partial d} \log \prod_i p\left(Y^i|\theta_d\right) \\
&= \sum_i \frac{\partial}{\partial d} \log p\left(Y^i|\theta_d\right) \\
&= \sum_i \frac{1}{p(Y^i|\theta_d)} \cdot \frac{\partial}{\partial d}\left(p\left(Y^i|\theta_d\right)\right) \\
&= \sum_i \frac{\sum_X \prod_a \left(f_a(X_a) + w_{X_a=x} \cdot d\right)}{\sum_{X|Y(X)=Y^i} \prod_a \left(f_a(X_a) + w_{X_a=x} \cdot d\right)} \cdot \frac{\partial}{\partial d}\left(\frac{\sum_{X|Y(X)=Y^i} \prod_a \left(f_a(X_a) + w_{X_a=x} \cdot d\right)}{\sum_X \prod_a \left(f_a(X_a) + w_{X_a=x} \cdot d\right)}\right) \\
&= \sum_i \frac{\sum_X \prod_a \left(f_a(X_a) + w_{X_a=x} \cdot d\right)}{\sum_{X|Y(X)=Y^i} \prod_a \left(f_a(X_a) + w_{X_a=x} \cdot d\right)} \cdot \\
&\quad \left(\frac{\sum_{X|Y(X)=Y^i} \sum_{a'} w_{X_{a'}=x} \prod_{a\neq a'} \left(f_a(X_a) + w_{X_{a'}=x} \cdot d\right) \cdot \sum_X \prod_a \left(f_a(X_a) + w_{X_{a'}=x} \cdot d\right)}{\left(\sum_X \prod_a \left(f_a(X_a) + w_{X_{a'}=x} \cdot d\right)\right)^2} - \right. \\
&\quad \left. \frac{\sum_X \sum_{a'} w_{X_{a'}=x} \prod_{a\neq a'} \left(f_a(X_a) + w_{X_{a'}=x} \cdot d\right) \cdot \sum_{X|Y(X)=Y^i} \prod_a \left(f_a(X_a) + w_{X_{a'}=x} \cdot d\right)}{\left(\sum_X \prod_a \left(f_a(X_a) + w_{X_{a'}=x} \cdot d\right)\right)^2}\right) \\
&= \sum_i \frac{\sum_{X|Y(X)=Y^i} \sum_{a'} w_{X_{a'}=x} \prod_{a\neq a'} \left(f_a(X_a) + w_{X_a} \cdot d\right)}{\sum_{X|Y(X)=Y^i} \prod_a \left(f_a(X_a) + w_{X_a} \cdot d\right)} - \frac{\sum_X \sum_{a'} w_{X_{a'}=x} \prod_{a\neq a'} \left(f_a(X_a) + w_{X_a} \cdot d\right)}{\sum_X \prod_a \left(f_a(X_a) + w_{X_a} \cdot d\right)}
\end{aligned}$$

It is not feasible to set this derivative equal to 0 to find the maximum value of d because there are so many higher-order terms in d.

Gradient descent can also be performed on the likelihood rather than the log likelihood. This will accentuate the gradient in the direction of large changes and ignore coordinates for which the absolute gradient is small. However, because of the many terms produced by taking the derivative of products mentioned above, it is only practical to optimize the likelihood with respect to a single data point at a time. Many of the algorithms developed in Section 3 will only require this capability, so the gradient with respect to the likelihood remains relevant.

$$\begin{aligned}
\frac{\partial P\left(Y|\theta\right)}{\partial f_{a'}(X_{a'} = x)} &= \frac{\partial}{\partial f_{a'}(X_{a'} = x)}\left(\frac{\sum_{X|Y(X)=Y} \prod_a f_a(X_a)}{\sum_X \prod_a f_a(X_a)}\right) \\
&= \frac{\left(\sum_{X|Y(X)=Y, X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)\right) \cdot \left(\sum_X \prod_a f_a(X_a)\right)}{\left(\sum_X \prod_a f_a(X_a)\right)^2} - \\
&\quad \frac{\left(\sum_{X|X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)\right) \cdot \left(\sum_{X|Y(X)=Y} \prod_a f_a(X_a)\right)}{\left(\sum_X \prod_a f_a(X_a)\right)^2} \\
&= \frac{\sum_{X|Y(X)=Y, X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_X \prod_a f_a(X_a)} - \\
&\quad \frac{\left(\sum_{X|X_{a'}=x} \left(\prod_{a\neq a'} f_a(X_a)\right)\right) \cdot \left(\sum_{X|Y(X)=Y} \prod_a f_a(X_a)\right)}{\left(\sum_X \prod_a f_a(X_a)\right)^2}
\end{aligned}$$

Of course, setting this derivative equal to zero yields the exact same extremum as when the corresponding calculation is performed for the log likelihood.

We thus find that gradient descent can be used in a variety of ways to maximize the log likelihood of an undirected graphical model. However, gradient ascent algorithms tends to be slow when the landscape in which they are operating is flat [27] We expect flat landscapes to be common when attempting to maximize the likelihood of undirected graphical models. For every hidden variable, each permutation of its values yields an identical probability distribution. In terms of the parameters θ, exchanging values y and z of hidden variable X_i corresponds to swapping $f_a(X_a \setminus X_i = x, X_i = y)$ with $f_a(X_a \setminus X_i = x, X_i = z)$ for all a and all $X_a \setminus X_i$. The likelihood maximization algorithm must then choose between all these equivalent representations of a single (locally) optimal probability distribution. We suspect that this will cause the landscape to be flat at points roughly equidistant between multiple such equivalent optimal distributions. We now turn to a class of algorithms which proceed more directly to the maximum and thus avoid some of these problems.

3 Expectation Maximization

As we saw on page 5, the log likelihood of a configuration of the observed variables requires a cumbersome marginalization, which prevents the logarithm from passing through to the product which constitutes $p(X|\theta)$. In contrast, the log likelihood can easily be calculated for full configurations

$$\log p(\mathbf{X}|\theta) = \sum_i \log \frac{1}{Z} \prod_a f_a(X_a^i) = -n \log Z + \sum_{i,a} \log f_a(X_a^i)$$

The expectation maximization (EM) algorithm is an iterative algorithm under which the log likelihood of the observed data ($\log p(\mathbf{Y}|\theta)$) provably converges to a stationary point, likely a local maximum, even though all operations are performed on a function of the total data (observed and hidden) [16, 31]. It therefore seems natural to apply the expectation maximization algorithm to maximize the log likelihood of the total data given the parameters. Each iteration of the expectation maximization algorithm consists of two parts: first, we calculate the expectation of the complete-data log likelihood given the old parameters and the observed data. Then, we maximize this value as a function of the parameters.

The expected value of the complete-data log likelihood is calculated as follows:

$$\begin{aligned}
E\left[\log p(\mathbf{X}) \mid \mathbf{Y}, \theta\right] &= \sum_{\mathbf{X}} p(\mathbf{X} \mid \mathbf{Y}, \theta) \log p(\mathbf{X} \mid \theta) \\
&= E\left[\log \prod_i \left(\frac{1}{Z} \prod_a f_a(X_a^i)\right) \mid \mathbf{Y}, \theta\right] \\
&= E\left[-n \log Z + \sum_{i,a} \log f_a(X_a^i) \mid \mathbf{Y}, \theta\right] \\
&= -nE\left[\log \left[\sum_X \prod_a f_a\left(X_a\right)\right] \mid \mathbf{Y}, \theta\right] + \sum_{\mathbf{X}} p(\mathbf{X} \mid \mathbf{Y}, \theta) \sum_{i,a} \log f_a(X_a^i) \\
&= -n \log \left[\sum_X \prod_a f_a\left(X_a\right)\right] + \sum_{i,X} p(X \mid Y^i, \theta) \sum_a \log f_a(X_a) \\
&= -n \log \left[\sum_X \prod_a f_a\left(X_a\right)\right] + \sum_{i,a,X_a} p(X_a \mid Y^i, \theta) \log f_a(X_a) \\
&= -n \log \left[\sum_X \prod_a f_a\left(X_a\right)\right] + \sum_{a,X_a} \left(\sum_i p\left(X_a \mid Y^i, \theta\right)\right) \log f_a(X_a)
\end{aligned}$$

In the transformations above, note that $\sum_{a,b} p(a,b) f(b) = \sum_b p(b) f(b)$.

3.1 Approximate EM update rule

To maximize the expression derived above with respect to each value of each relation, we set the partial derivative with respect to $f_a(X_a = x)$ equal to zero:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial f_{a'}(X_{a'}=x)} E\left[\log p(\mathbf{X}) \mid \mathbf{Y}, \theta\right] \\
&= \frac{\partial}{\partial f_{a'}(X_{a'}=x)} \left(-n \log \left(\sum_X \prod_a f_a\left(X_a\right) \right) + \sum_{a, X_a} \left(\sum_i p\left(X_a^i \mid Y^i, \theta\right) \right) \cdot \log f_a(X_a) \right) \\
&= -n \frac{\sum_{X \mid X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)}{\sum_X \prod_a f_a(X_a)} + \frac{\sum_i p(X_{a'}^i = x \mid Y^i, \theta)}{f_{a'}(X_{a'}=x)} \\
f_{a'}(X_{a'}=x) &= \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X \mid X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)} \cdot \frac{1}{n} \sum_i p(X_{a'}^i = x \mid Y^i, \theta)
\end{aligned}
$$

The partition function $Z = \sum_X \prod_a f_a(X_a)$ in the numerator of the final expression includes the term $f_{a'}(X_{a'}=x)$. We can rewrite the partition function as

$$
\begin{aligned}
Z &= \sum_X \left(f_{a'}(X_{a'}) \cdot \prod_{a \neq a'} f_a(X_a) \right) \\
&= \left(\sum_{X \mid X_{a'}=x} f_{a'}(X_{a'}=x) \cdot \prod_{a \neq a'} f_a(X_a) \right) + \left(\sum_{X \mid X_{a'} \neq x} f_{a'}(X_{a'}) \cdot \prod_{a \neq a'} f_a(X_a) \right)
\end{aligned}
$$

Since most terms in Z do not correspond to $X_{a'} = x$ and so do not include $f_{a'}(X_{a'}=x)$, the contribution of $f_{a'}(X_{a'}=x)$ is small so long as all entries of a single relation have approximately equal values. Thus, when calculating Z we can make the approximation that $f_{a'}^{t+1}(X_{a'}=x) \approx f_{a'}^t(X_{a'}=x)$, which corresponds to $Z^{t+1} \approx Z^t$. The update then becomes

$$
\begin{aligned}
f_{a'}^{t+1}(X_{a'}=x) &= \frac{Z^{t+1}}{\sum_{X \mid X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)} \cdot \frac{1}{n} \sum_i p(X_{a'}^i = x \mid Y^i, \theta^t) \\
&\approx \frac{Z^t}{\sum_{X \mid X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)} \cdot \frac{1}{n} \sum_i p(X_{a'}^i = x \mid Y^i, \theta^t) \\
&\approx f_{a'}^t(X_{a'}=x) \cdot \frac{\left\langle p(X_{a'}=x | Y^i, \theta^t) \right\rangle}{p(X_{a'}=x | \theta^t)}
\end{aligned}
$$

We shall refer to this update rule as the approximate EM update. The approximation $Z^{t+1} \approx Z^t$ becomes an equality if $f_{a'}^{t+1}(X_{a'}=x) = f_{a'}^t(X_{a'}=x)$, so the parameters are guaranteed to be an extremum of the likelihood if they converge. In the update equations below, all probabilities conditioned on θ refer to θ^t unless otherwise specified.

We have no a priori reason to believe that the extremum found by setting the partial derivative of the log likelihood equal to zero will be a maximum of the likelihood rather than a minimum. This is all the more disturbing since, with these approximations, we are only able to find a single extremum. If this point were a maximum half the time and a minimum the other half of the time, we might expect to make little progress. Fortunately, the approximate extremum returned by the above formula on each iteration is always in the direction of increasing likelihood.

$$
\begin{aligned}
\left(f_{a'}(X_{a'}=x)^{t+1} - f_{a'}^t(X_{a'}=x)\right) / \frac{\partial \log p(\mathbf{Y}|\theta)}{\partial f_{a'}(X_{a'}=x)} &= \frac{f_{a'}^t(X_{a'}=x) \cdot \frac{\left\langle p(X_{a'}=x|Y^i,\theta) \right\rangle}{p(X_{a'}=x|\theta)} - f_{a'}^t(X_{a'}=x)}{\frac{n}{f_{a'}^t(X_{a'}=x)} \left(\left\langle p(X_{a'}=x|Y^i,\theta) \right\rangle - p(X_{a'}=x|\theta) \right)} \\
&= \frac{f_{a'}^t(X_{a'}=x)^2}{n \cdot p(X_{a'}=x|\theta)} \\
&> 0
\end{aligned}
$$

Surprisingly, this expectation maximization update is the same as that yielded by the iterative proportional fitting procedure (IPFP) [50, 51], as can be seen by rewriting the update as follows:

$$
f_{a'}^{t+1}(X_{a'}=x) = f_{a'}^t(X_{a'}=x) \cdot \frac{\left\langle p(X_{a'}^i=x|Y^i,\theta) \right\rangle}{p(X_{a'}=x|\theta)}
$$

IPFP cycles through the parameters, multiplying them by a term designed to make the marginal probability of a configuration of a factor node match the observed probability of the configuration. Since this procedure depends upon knowing the observed probability given the data, it normally requires that all variables be observed. Thus, the EM method extends the IPFP update to undirected graphical models with hidden variables. Wiegerinck and Heskes previously derived the EM extension of IPFP [51]. Our derivation proceeds via a different method, and is perhaps more rigorous.

Both empirically and logically, the entries of a relation rarely have approximately equal values. Such a state would imply that all configurations of visible and hidden variables are equally likely, and thus the modeled probability distribution is uniform. Consider a relation between two visible variables and one hidden variable, for which only one value of the hidden variable has high marginal probability for each combination of the two visible variables. We expect this to be a common situation, since if many values of the hidden variable have large marginal probability for each choice of the visible variables, it implies that the visible variables connected to the relation in question have no impact on the probability distribution of the rest of the visible variables. The partition function can be factored as follows:

$$Z = \sum_{X_{a'}} f_{a'}(X_{a'}) \sum_{X|X_{a'}} \prod_{a \neq a'} f_a(X_a)$$

Assuming that each combination of the two visible variables is approximately equally likely (as will be the case in our example problem in Section 3.4), there will be one entry of relation a' that is significantly greater than 0 for each value of the hidden variable. Thus, the factorization of Z given above will have $n_1 \times n_2$ terms significantly greater than 0, where n_i is the number of values which visible variable i may assume. As the n_i grow small, there are fewer terms, so changing one term will have a proportionally larger effect on Z. It is thus possible that in undirected graphical models where the number of values for each variable is small, such as Boltzmann machines (discussed in Section 4), the approximation that $Z^{t+1} \approx Z^t$ will not be accurate and the simple update rule given above will not effectively maximize the log likelihood on each step. Note however that the extremum of the likelihood will still be achieved if the parameters reach a fixed point.

3.2 Exact EM update rule

If we don't choose to ignore the term $f_{a'}(X_{a'} = x)$ in Z, then we can explicitly subtract it from Z and obtain the following equation for the parameter update:

$$\begin{aligned}
f_{a'}^{t+1}(X_{a'} = x) &= \frac{1}{n} \cdot \frac{\sum_X \prod_a f_a(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)} \sum_i p(X_{a'}^i = x | Y^i, \theta) \\
&= \frac{1}{n} \cdot \frac{\sum_X \prod_a f_{a'}^t(X_a) - f_{a'}^t(X_{a'} = x) \cdot \sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a^t\left(X_a\right) \right)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)} \sum_i p(X_{a'}^i = x | Y^i, \theta) \\
&\quad + \frac{1}{n} \cdot \frac{f_{a'}^{t+1}(X_{a'} = x) \cdot \sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a^t\left(X_a\right) \right)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a\left(X_a\right) \right)} \sum_i p(X_{a'}^i = x | Y^i, \theta) \\
&= \frac{\frac{1}{n} \cdot \left(\frac{\sum_X \prod_a f_{a'}^t(X_a)}{\sum_{X|X_{a'}=x} \left(\prod_{a \neq a'} f_a(X_a) \right)} - f_{a'}^t(X_{a'} = x) \right) \sum_i p(X_{a'}^i = x | Y^i, \theta)}{1 - \frac{1}{n} \cdot \sum_i p(X_{a'}^i = x | Y^i, \theta)} \\
&= \frac{f_{a'}^t(X_{a'} = x) \left(\frac{1}{p(X_{a'} = x|\theta)} - 1 \right) \cdot \left\langle p(X_{a'} = x | Y^i, \theta) \right\rangle}{1 - \left\langle p(X_{a'} = x | Y^i, \theta) \right\rangle}
\end{aligned}$$

We shall refer to this update rule as the exact EM update.

Once again, this extremum is always in the direction of increasing likelihood.

$$\begin{aligned}
\left(f_{a'}^{t+1}(X_{a'}=x) - f_{a'}^{t}(X_{a'}=x)\right) / \frac{\partial \log p(\mathbf{Y}|\theta)}{\partial f_{a'}(X_{a'}=x)} &= \frac{\frac{f_{a'}^{t}(X_{a'}=x)\left(\frac{1}{p(X_{a'}=x|\theta)}-1\right)\cdot\langle p(X_{a'}=x|Y^i,\theta)\rangle}{1-\langle p(X_{a'}=x|Y^i,\theta)\rangle} - f_{a'}^{t}(X_{a'}=x)}{\frac{n}{f_{a'}^{t}(X_{a'}=x)}\left(\langle p(X_{a'=x}|Y^i,\theta)\rangle - p(X_{a'}=x|\theta)\right)} \\
&= \frac{f_{a'}^{t}(X_{a'}=x)^2}{n\cdot(1-\langle p(X_{a'}=x|Y^i,\theta)\rangle)} \cdot \frac{\frac{\langle p(X_{a'}=x|Y^i,\theta)\rangle}{p(X_{a'}=x|\theta)} - \langle p(X_{a'}=x|Y^i,\theta)\rangle - 1 + \langle p(X_{a'}=x|Y^i,\theta)\rangle}{\langle p(X_{a'}=x|Y^i,\theta)\rangle - p(X_{a'}=x|\theta)} \\
&= \frac{f_{a'}^{t}(X_{a'}=x)^2}{n\cdot p(X_{a'}=x|\theta)\cdot(1-\langle p(X_{a'}=x|Y^i,\theta)\rangle)} \cdot \frac{\langle p(X_{a'}=x|Y^i,\theta)\rangle - p(X_{a'}=x|\theta)}{\langle p(X_{a'}=x|Y^i,\theta)\rangle - p(X_{a'}=x|\theta)} \\
&= \frac{f_{a'}^{t}(X_{a'}=x)^2}{n\cdot p(X_{a'}=x|\theta)\cdot(1-\langle p(X_{a'}=x|Y^i,\theta)\rangle)} \\
&> 0
\end{aligned}$$

The magnitude of the change in parameter suggested by the exact EM update is a factor of $\frac{1}{1-\langle p(X_{a'}=x|Y^i,\theta)\rangle}$ as large as that suggested by the approximate EM update. The two updates will thus be very similar early in learning, when the probability of the all observed configurations is small. Additionally, the two updates may differ more in systems with only a few values for each hidden variable, since then the probability of most configurations is significantly different from zero. We have not seen a significant difference in performance between the approximate and exact EM updates in any of our experiments. Except where explicitly noted, all experimental results described in Section 3.5 were obtained with the exact EM update.

We will assume that $f_a^{\theta^{t+1}}(X_a) \approx f_a^{\theta^t}(X_a)$ in the above expression and will generally update multiple parameters without recalculating $Z = \sum_X \prod_a f_a(X_a)$. This assumption is accurate if either only one parameter value is updated after each expectation calculation, or in the limit of small updates. However, we do recalculate $\text{Norm} = \sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a\left(X_a\right)\right)$ when necessary. Note that the norm does not depend upon the relation being updated. Empirically, recalculating the norm before updating each relation does not seem to speed convergence of the algorithm. We have not tried recalculating Z between each parameter update since this would be prohibitively computationally expensive.

Just as in the case of gradient maximization, we can attempt to maximize along a vector in parameter space which includes multiple components. Here, however, expectation maximization will save us from an explosion of terms.

$$\begin{aligned}
\theta_d &= [f_a(X_a) + w_{X_a}\cdot d] \\
\frac{\partial}{\partial d}E\left[\log p(\mathbf{X}) \mid \mathbf{Y}, \theta_d\right] &= \frac{\partial}{\partial d}\left(-n\log\left(\sum_X\prod_a\left(f_a\left(X_a\right)+w_{X_a}\cdot d\right)\right) + \sum_{a,X_a}\left(\sum_i p\left(X_a^i|Y^i,\theta_d\right)\right)\cdot\log\left(f_a(X_a)+w_{X_a}\cdot d\right)\right) \\
&= -n\frac{\sum_{a',X_{a'}} w_{X_{a'}}\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'}\left(f_a\left(X_a\right)+w_{X_a}\cdot d\right)\right)}{\sum_X\prod_a\left(f_a(X_a)+w_{X_a}\cdot d\right)} + \sum_{a',X_{a'}} w_{X_{a'}}\frac{\sum_i p(X_{a'}^i=x|Y^i,\theta_d)}{f_{a'}(X_{a'}=x)+w_{X_{a'}}\cdot d}
\end{aligned}$$

Assuming that d is small, we will ignore its effect in the first term and perform a first-order Taylor expansion of the second term.

$$\begin{aligned}
\frac{\partial}{\partial d}E\left[\log p(\mathbf{X}) \mid \mathbf{Y}, \theta_d\right] &= -n\frac{\sum_{a',X_{a'}} w_{X_{a'}}\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a\left(X_a\right)\right)}{\sum_X\prod_a f_a(X_a)} + \\
&\quad \sum_{a',X_{a'}} w_{X_{a'}}\cdot\sum_i\frac{p(X_{a'}^i=x|Y^i,\theta_d)}{f_{a'}(X_{a'}=x)}\left(1-\frac{w_{X_{a'}}}{f_{a'}(X_{a'}=x)}\cdot d\right)
\end{aligned}$$

Setting this derivative equal to zero to find the extremum, we obtain

$$\begin{aligned}
d &= \frac{-n\frac{\sum_{a',X_{a'}} w_{X_{a'}}\sum_{X|X_{a'}=x}\left(\prod_{a\neq a'} f_a(X_a)\right)}{\sum_X\prod_a f_a(X_a)} + \sum_{a',X_{a'}} w_{X_{a'}}\cdot\sum_i\frac{p(X_{a'}^i=x|Y^i,\theta_d)}{f_{a'}(X_{a'}=x)}}{\sum_{a',X_{a'}} w_{X_{a'}}^2\cdot\sum_i\frac{p(X_{a'}^i=x|Y^i,\theta_d)}{(f_{a'}(X_{a'}=x))^2}} \\
&= \frac{-\sum_{a',X_{a'}} w_{X_{a'}}\frac{p(X_{a'}=x|\theta_d)}{f_{a'}(X_{a'})} + \sum_{a',X_{a'}} w_{X_{a'}}\cdot\left\langle\frac{p(X_{a'}^i=x|Y^i,\theta_d)}{f_{a'}(X_{a'}=x)}\right\rangle}{\sum_{a',X_{a'}} w_{X_{a'}}^2\cdot\left\langle\frac{p(X_{a'}^i=x|Y^i,\theta_d)}{(f_{a'}(X_{a'}=x))^2}\right\rangle}
\end{aligned}$$

We have not yet tested this update rule. However, if the first-order Taylor expansion is a sufficiently accurate approximation, this update rule could allow the likelihood to be optimized using the conjugate gradient method, which is often found to be more efficient than optimization along an arbitrary set of coordinates [38].

3.3 Belief propagation

All of the EM updates are composed of the terms $p(X_{a'} = x|\theta)$ and $p(X_{a'} = x|Y, \theta)$. These, in turn, consist of sums of products of the relations. When the factor graph is a tree, we can calculate these probabilities exactly using belief propagation [2, 54]. Similarly, the value of the norm of relation a and configuration $X_a = x$ is equal to the product of the messages received by relation a for $X_a = x$ in the belief propagation algorithm when the observed variables are set equal to 1, while Z is equal to the sum of the norms calculated in this way multiplied by the value of the relation for each configuration. Note that while a single value of Z can be used to calculate $p(X_{a'} = x|Y, \theta)$ at all relations and for all configurations at a relation, a separate normalizing factor is needed for each relation and local configuration (a, X_a).

Even when the factor graph contains loops, it is possible to use belief propagation to approximate the probabilities, norms, and partition functions [35]. Belief propagation minimizes the Bethe free energy on loopy factor graphs [55]. Alternatively, generalized belief propagation algorithms can be used which minimize the more accurate Kikuchi free energy. Since loopy belief propagation is more time consuming and less accurate than belief propagation on trees, we restrict ourselves to learning undirected graphical models on trees in this thesis.

The connectivity pattern of the cortex, in contrast, is similar to a hierarchical grid [3], and so is extremely loopy. Many of these loops, such as those formed by the extensive connections within a single cortical column, have a small diameter. Such short loops are believed to be more disruptive to belief propagation than large loops [35]. Regardless of the correct interpretation of spikes in the cortex, it is well known that information can propagate from sensory systems through analytic and decision-making faculties and effect motor responses within 150-180 milliseconds, whereas a single neuron requires a few milliseconds to release neurotransmitter vesicles from axonal boutons after receiving dendritic input. Thus, only a few dozen synapses could be activated in the time between stimulus and response. Many of these synapses must transmit the signal up the neural hierarchy. In the case of the visual system, input must travel from retina (including photoreceptors, bipolar cells, and retinal ganglion cells) through the lateral geniculate nucleus to V1 and then on through V4 and IT to the frontal cortex and finally to the motor areas [15]. Thus, there is only enough time for information to propagate through a few recurrent cycles in each region in the fastest human responses. If loopy belief propagation is performed by the cortex, the number of cycles required for convergence. This suggests that loopy belief propagation may converge particularly quickly on topologies similar to that in the cortex. We plan to investigate this possibility in the future.

3.4 An example problem

In the following sections, we will test various undirected graphical model learning rules using an extension of the parity problem to non-binary values. When testing a network with n inputs, we will randomly select n real numbers r_i, $0 \leq r_i \leq 1$ such that $\sum_{i=1}^{n} r_i \mod 1 = 0$. This problem is much more difficult than those faced by the brain because the marginal probability of any subset of the variables is uniform; order only emerges when all of the inputs are known. Thus, local nodes in the graphical model cannot learn any correlations independently of the rest of the network, and the parameters of the relations must be trained as a unitary whole. In contrast, sensory tasks have substantial local structure. To use a visual analogy, the parity problem roughly corresponds to identifying random dot patterns, whereas the images the human visual system is most adept at processing contain continuous lines and surfaces.

As an intuitive demonstration of the difficulty of this problem, consider the following two collections of ten data points, where each column constitutes a single configuration of six input variables:

0.0439	0.0928	0.1901	0.6927	0.6756	0.4508	0.2324	0.0784	0.9943	0.5915
0.0272	0.0353	0.5869	0.0841	0.6992	0.7159	0.8049	0.6408	0.4398	0.1197
0.3127	0.6124	0.0576	0.4544	0.7275	0.8928	0.9084	0.1909	0.3400	0.0381
0.0129	0.6085	0.3676	0.4418	0.4784	0.2731	0.2319	0.8439	0.3142	0.4586
0.3840	0.0158	0.6315	0.3533	0.5548	0.2548	0.2393	0.1739	0.3651	0.8699
0.6831	0.0164	0.7176	0.1536	0.1210	0.8656	0.0498	0.1708	0.3932	0.9342

0.0196	0.7095	0.6822	0.3784	0.8998	0.3420	0.3093	0.5466	0.9568	0.2714
0.6813	0.4289	0.3028	0.8600	0.8216	0.2897	0.8385	0.4449	0.5226	0.2523
0.3795	0.3046	0.5417	0.8537	0.6449	0.3412	0.5681	0.6946	0.8801	0.8757
0.8318	0.1897	0.1509	0.5936	0.8180	0.5341	0.3704	0.6213	0.1730	0.7373
0.5028	0.1934	0.6979	0.4966	0.6602	0.7271	0.7027	0.7948	0.9797	0.1365
0.5850	0.1739	0.6246	0.8178	0.1555	0.7659	0.2110	0.8979	0.4877	0.7267

The first data set is random, whereas the second data set is drawn from our test parity problem. Distinguishing between the two without prior knowledge of their defining properties is well beyond the author's capabilities.

Although the parity problem is extremely difficult, it does lend itself to identification by an undirected graphical model arranged as a tree in which each variable node has degree at most two. That is, pairs of relations are connected by a single hidden variable. For the sake of computational simplicity, we restrict ourselves to relations of degree three. Unless otherwise noted, all tests are performed on trees with four relations and six visible variables. A depiction of such a tree is shown in Figure 1. If the inputs are presented to the leaves of the tree, then each relation node should set each hidden variable to which it is connected equal to the sum of its other two inputs (or one minus the sum of the other two inputs).

Our undirected graphical models contain only discrete variables, so the parity problem cannot be implemented exactly. In all tests, the visible variables will have 30 possible values and the hidden variables will have 10 possible values. To simplify the interpretation of the visible variables, the ith value will be associated with the number $\frac{n}{30}$. After selecting a number for each visible variable as described above, the activity associated with each value of the visible variables is set according to a Gaussian function of width $\sqrt{\frac{1}{30}}$ and height $\sqrt{\frac{3}{10}}$ centered on the input. This allows generalization between nearby values. (Qualitatively similar results are obtained with a height of 1.)

3.5 Implementation considerations

There are many sets of relations $f_a(X_a)$ which yield the same probability distribution. For instance, because of the action of the partition function, multiplying any relation by a constant does not change the probability distribution. This implies that there are no bounds on the relations, and they can easily grow larger or smaller than computer datatypes can accurately represent. For this reason, we normalize all relations after each update except when learning by gradient ascent, which is traditionally performed without normalization [12]. We have had success multiplying each entry $f_{a'}(X_{a'} = x)$ by

$$\frac{1}{\left(\sum_X \prod_a f_a(X_a)\right)^{\frac{1}{n}}},$$

which assures that $Z = \sum_X \prod_a f_a(X_a) = 1$, and

$$\prod_{v \in N(a)} \left(\frac{\sqrt[|N(v)|]{\prod_{a' \in N(v)} \frac{1}{|X_{a'}|X_v=x|} \sum_{X_{a'}|X_v=x} f_{a'}(X_{a'})}}{\frac{1}{|X_a|X_v=x|} \sum_{X_a|X_v=x} f_a(X_a)} \right),$$

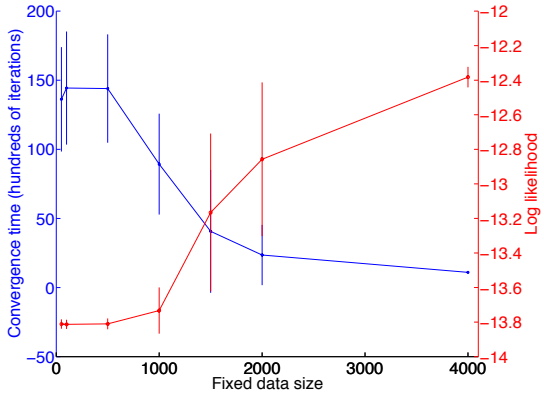
where $N(a)$ denotes the variables adjacent to relation a in the factor graph and $N(v)$ indicates the relations adjacent to variable v in the factor graph. This normalization assures that, for each value of a hidden variable, the average parameter associated with that hidden variable is equal across the relations connected to it. As mentioned above, the first normalization does not affect the probability distribution because the same term appears in the partition function. The second normalization leaves the probability distribution unchanged because each $\prod_a f_a(X_a)$ includes all of the relations that are neighbors of each hidden variable exactly once, so the product of the normalization constants is 1.

Since both the approximate and exact EM updates include the previous value of $f_a(X_a = x)$ as a multiplicative factor, if $f_a(X_a = x)$ ever becomes equal to 0, it can never recover. We thus impose a minimum value of 10^{-8} on each parameter. This does not seem to impair the quality of the final solutions, and prevents small values from becoming locked in prematurely.

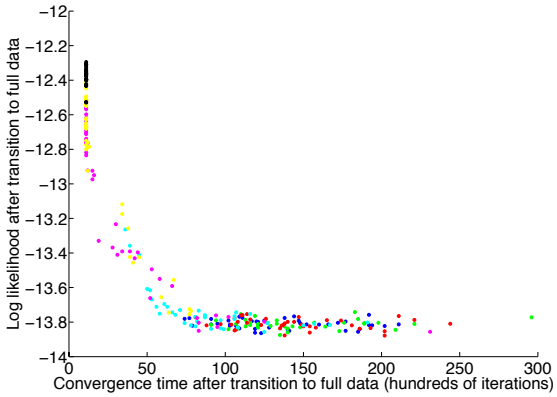
For each update, the EM updates require the probability of each relation configuration given the current parameters for each data point ($p(X_a = x|Y^i, \theta) \ \forall \ a, x, i$). However, we would like to perform updates progressively as new data points become available. Moreover, to save memory and processing time, we would like to avoid storing all past data points explicitly and refrain from recalculating the marginal probabilities induced by all previous data points using the newly updated parameters. This is necessary both for online learning, where all data points are not initially available, and for problems with an infinite number of possible data points, as is the case in the parity

problem (Section 3.4). In these latter cases, we do not wish to restrict ourselves to a preselected subset of the possible data points, nor is it feasible to recompute the necessary probabilities for each previously used data point.

Empirically, the probability distributions learned when the system is trained on a few hundred fixed data points are not the same as when each update is based on a few hundred data points, but the data points used are changed continuously (via the decaying update rule described below). However, there is a critical point above which it is possible to learn probability distributions similar to the true distribution using a fixed data set. This is visible in the log likelihood after training on a fixed data set versus the log likelihood after training on the full data set, as well as in the time required to converge on the full data set after training on the fixed data set (Figure 3). In this and all following figures, successive data sets are plotted with the colors blue, green, red, cyan, magenta, yellow, and black.



Time at which the log likelihood attains 95% of its maximum value versus the final log likelihood (blue, left) and initial log likelihood (red, right) after transitioning from a fixed data set to the full data set. Error bars indicate the standard deviation of the data.



Time at which the log likelihood attains 95% of its maximum value after transitioning from a fixed data set to a full data set versus the initial log likelihood after the transition. Points correspond to data sets of size 50 (blue), 100 (green), 500 (red), 1000 (cyan), 1500 (magenta), 2000 (yellow), and 4000 (black).

Figure 3: Effect of small fixed data sets on convergence time and log likelihood

Both the exact and the linearized update rules only depend upon the observed data through the term

$$\frac{1}{n}\sum_i p(X^i_{a'} = x|Y^i, \theta^t) = \left\langle p(X_{a'} = x|Y^i, \theta^t)\right\rangle_i$$

Thus, if we can estimate the average marginal probability of a relation's arguments assuming a particular configuration given the observed data, we can calculate the new value of the relation at that configuration. It would be simple enough to sum all previous estimates of $p(X^i_{a'} = x|Y^i, \theta)$, but if we are learning progressively, then most of the values at any point in time will have been calculated using old values of the parameters. The older the value of $p(X^i_{a'} = x|Y^i, \theta^t)$, the more the relation will have changed since the marginal probability was computed and the greater the error of the term. Thus, we perform a weighted summation, such that the contribution of each term decays with time. We will parameterize this summation by a learning rate λ, which ranges between 0 (slow learning) and 1 (fast learning). Note that

$$\sum_{i=0}^{\infty} a^i = \frac{1}{1-a}$$

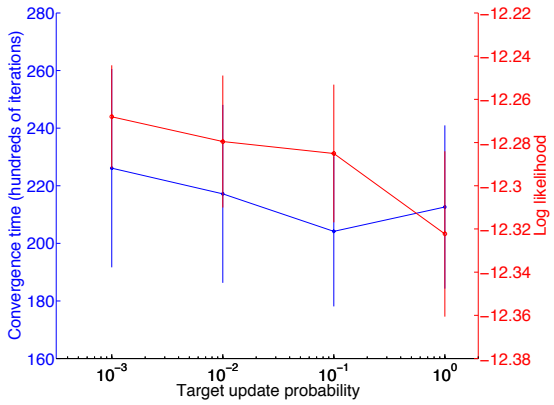
The update rule

$$\left\langle p_{X_{a'}=x}\right\rangle_t = \lambda\left\langle p_{X_{a'}=x}\right\rangle_{t-1} + (1-\lambda)p(X_{a'} = x|Y^i, \theta^t)$$

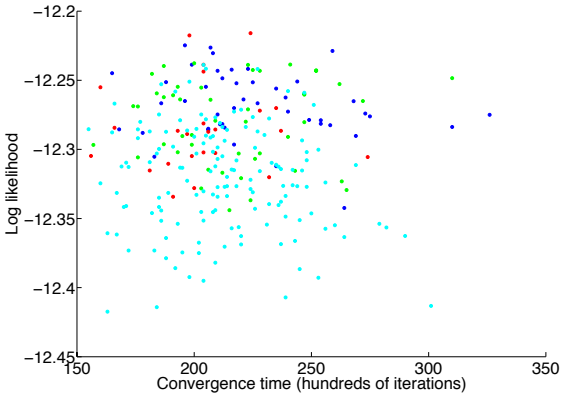
assures that the contribution of each term decays exponentially with time, and that the total sum is properly normalized.

In general, we recalculate the estimated probability of each configuration of the arguments of each relation $\left\langle p_{X_{a'}} = x|Y^i, \theta^t\right\rangle_i$ after each new data point is presented, and use this to update each parameter using the exact EM update rule described above. Since many data points must be sampled to estimate the likelihood of the data given a set of parameters but the parameters are changed in response to each data point, the parameters are always very close to the optimal values determined by the update rule given the present estimates of the marginal probabilities.

The convergence properties of the algorithm are unchanged if only a fraction of the parameters are updated in response to each data point (Figure 4), or if multiple data points are collected for each synchronous update of all parameters (Figure 5). In Figure 4, the probability of updating a parameter was increased if other parameters had been updated many more times, so the actual update probability was slightly higher than that listed. Note that the green points are obscured by the magenta points. With a learning rate of 0.0489 and a batch of size 10 or a learning rate of 0.3942 and a batch of size 100, probabilities are updated at approximately the same rate as with a batch of size 1 and a learning rate of 0.005. As can be seen from Figure 5, the convergence properties are almost identical in these cases. Lower learning rates result in slower convergence, as we shall see in more detail below. The cyan cluster corresponding to a batch of size 100 and a learning rate of 0.005 does not converge within the alloted time.



Target parameter update probability versus the time at which the log likelihood attains 95% of its maximum value (blue, left) and the final log likelihood (red, right). Error bars indicate the standard deviation of the data.



Time at which the log likelihood attains 95% of its maximum value versus the final log likelihood. Points correspond to the target parameter update probabilities 0.001 (blue), 0.01 (green), 0.1 (red), and 1 (cyan).

Figure 4: Effect of parameter update probability on convergence time and log likelihood

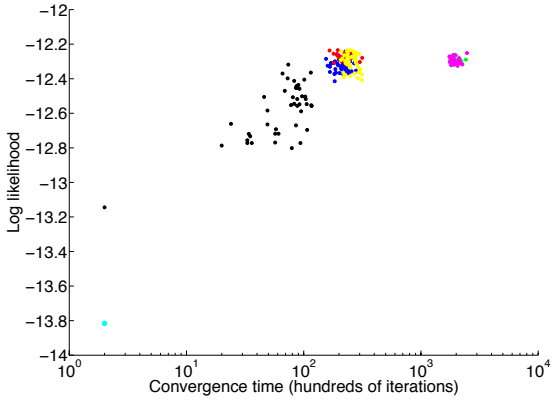
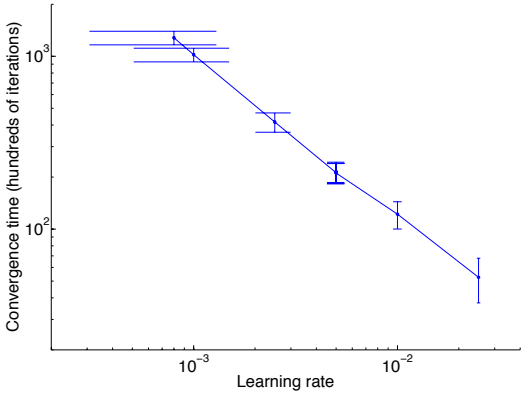


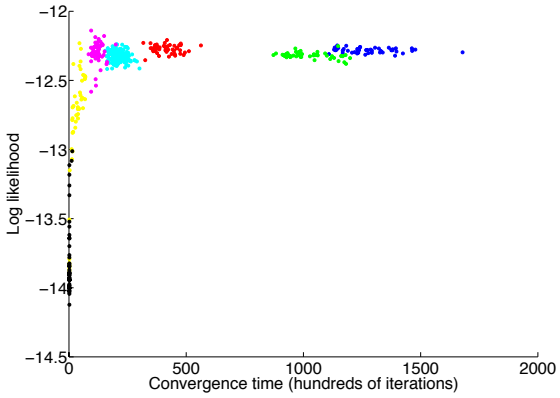
Figure 5: Time at which the log likelihood attains 95% of its maximum value versus the final log likelihood for various batch parameters. Points correspond to batch size (b) and learning rate (lr) [b 1, lr 0.005] (blue), [b 10, lr 0.005] (green), [b 10, lr 0.0489] (red), [b 100, lr 0.005] (cyan), [b 100 lr 0.05] (magenta), [b 100, lr 0.3942] (yellow), [b 100, lr 1] (black).

We might expect that learning would progress more quickly for larger learning rates. Consider two learning rates x and y. The time required for an old marginal probability to decay by a factor a with these two rates is m and n respectively, where $x^m = y^n = a$. Thus, $\frac{m}{n} = \log(y - x)$. We in fact see the expected effect experimentally in Figure 6 (left), but the relationship between learning rate and convergence time appears to be polynomial rather than logarithmic. When the learning rate is too large, convergence is very fast but performance decays, as in Figure 6 (right). Overall, however, the performance is robust with respect to the learning rate.

By symmetry, it is apparent that if all of the entries of the relations are equal, then all of the updates will be equal



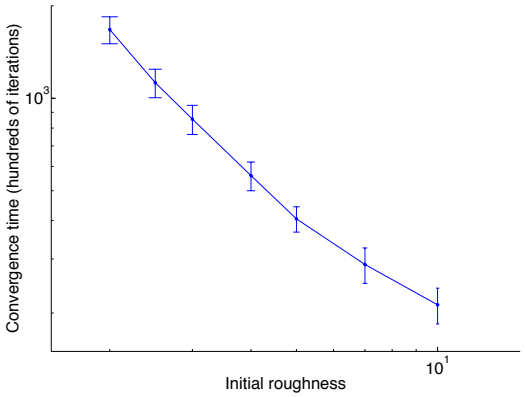
Learning rate versus time at which the log likelihood attains 95% of its maximum value. Error bars indicate the standard deviation of the data.



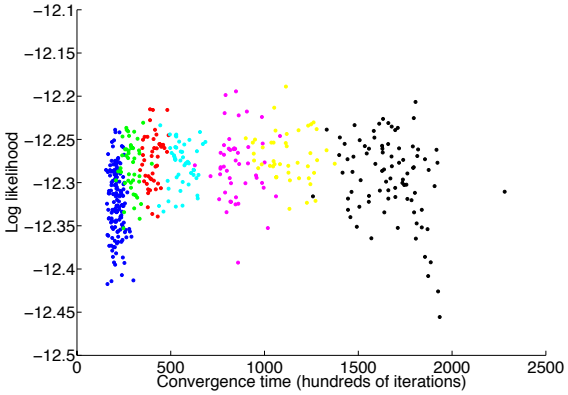
Time at which the log likelihood attains 95% of its maximum value versus the final log likelihood. Points correspond to the learning rates 0.0008 (blue), 0.001 (green), 0.0025 (red), 0.005 (cyan), 0.01 (magenta), 0.025 (yellow), and 0.05 (black)

Figure 6: Effect of learning rate on convergence time and log likelihood

and learning will not occur. Of course, this is an unstable equilibrium, and only a small perturbation from perfect equality is necessary to break the symmetry and allow learning to occur. However, the magnitude of the updates and thus the rate of learning is dependent upon the difference between the relation entries, even when they are not all equal. Initial relation parameters were selected according to the formula e^{-rx}, where x is a random variable uniformly chosen from the interval $[0, 1)$ and r is a constant referred to as the roughness factor. As can be seen from Figure 7 (left), there is a polynomial relationship between the roughness factor and the convergence time. Future investigations will determine to what degree this is due to the varying initial maximum-to-minimum parameter ratio as opposed to the nonuniformity of the parameter distribution. Figure 7 (right) shows that as the roughness factor increases, the log likelihood after convergence decreases.



Roughness factor versus time at which the log likelihood attains 95% of its maximum value. Error bars indicate the standard deviation of the data.



Time at which the log likelihood attains 95% of its maximum value versus the final log likelihood. Points correspond to the roughness factors 10 (blue), 7 (green), 5 (red), 4 (cyan), 3 (magenta), 2.5 (yellow), and 2 (black)

Figure 7: Effect of roughness factor on convergence time and log likelihood

Convergence is attained more rapidly when a small fixed set of data points are used, rather than the entire data set. As we saw previously, pretraining on a fixed data set also speeds convergence on the full data set in a manner similar to increasing the roughness of the initial parameters. The decrease in the convergence time after pretraining on a fixed data set is in fact greater than the time required to reach convergence on the fixed data set (Figure 8), so pretraining on a small fixed data set can reduce overall training time.

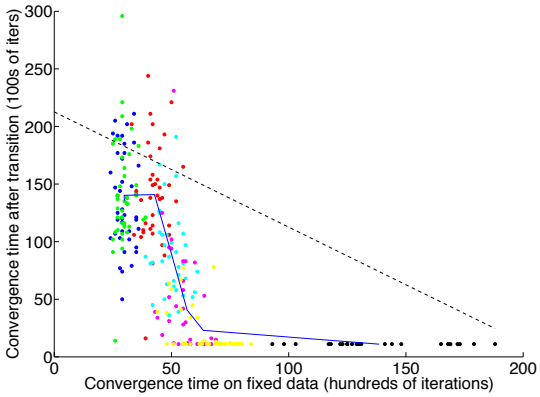


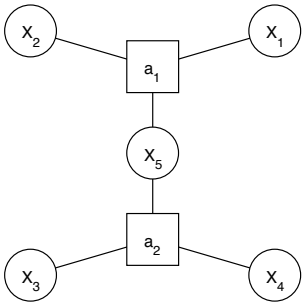
Figure 8: Time at which the log likelihood attains 95% of its maximum value (convergence time) on the fixed data set versus convergence time on the full data set after pretraining on the fixed data set. Points correspond to the fixed data set sizes 50 (blue), 100 (green), 500 (red), 1000 (cyan), 1500 (magenta), 2000 (yellow), and 4000 (black). The solid blue line indicates the average for each fixed data set size. The dashed black line indicates the average convergence time without pretraining (212.6259 with a standard deviation of 28.3417)

Although we have not been able to find any performance difference between the exact EM update and the linearized EM update, both exhibit performance far superior to gradient-based updates. We examined both gradient updates using the probability accumulation scheme described above (acc), and gradient updates based exclusively upon the present data point (no acc), under a wide range of learning rates. Gradient updates did not converge within a reasonable amount of time on the four relation / six visible variable parity problem, nor on the three relation / five visible variable parity problem. Thus, we examined their properties on the two relation / four visible variable parity problem depicted in figure 9 (left). As can be seen from Figure 9 (right), these techniques were uniformly slower and less accurate than EM updates. Simulations with no probability accumulation and a learning rate of 0.001 (blue circles) failed to converge, hence their small log likelihood.

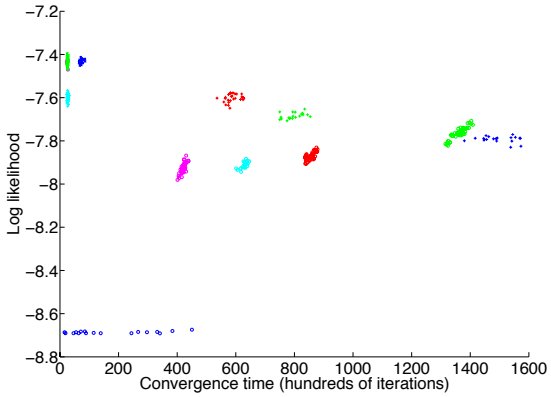
A similar technique can improve performance when there are a finite number of data points, but Gibbs sampling is used to estimate the probability of each configuration. Generally, Gibbs sampling is restarted from scratch every time the parameters are changed. However, the probability distribution is continuous in the parameters, so small parameter changes generally lead to small changes in the probability distribution. It is thus both safe and more efficient to use the old probability distribution as a basis for calculating the new distribution after a parameter update, both in that the current state of all the variables can be retained rather than restarting the Gibbs sampler from a random state, and in that the probability of each configuration can be calculated as a weighted historical average, using the old probabilities before the parameter change as a starting point rather than reinitializing from zero. This will add a bit of bias to the calculation, but will substantially decrease the variance.

4 Boltzmann machines

Boltzmann machines are undirected graphical models in which each variable has exactly two states and each relation has degree one or two. Unlike the graphical models considered in our example problem, in which variables always had degree two (i.e., they were connected to exactly two relations), variables in Boltzmann machines are frequently connected to every other variable; since a relation can only connect two variables, each variable must be connected to $n_v - 1$ relations, where n_v is the number of variables. In Boltzmann machines the variables assume values in the range $S_i \in \{-1, 1\}$. The probability distribution of the Boltzmann machine is defined by it's energy function $H(\mathbf{S})$,



Factor graph for the two relation / four visible variable parity problem.



Time at which the log likelihood attains 95% of its maximum value versus the final log likelihood for various gradient updates. Points correspond to the learning rules [acc, lr 0.00025] (blue +), [acc, lr 0.0005] (green +), [acc, lr 0.00075] (red +), [acc, lr 0.001] (cyan +), [no acc, lr 0.001] (blue o), [no acc, lr 0.005] (green o), [no acc, lr 0.0075] (red o), [no acc, lr 0.01] (cyan o), [no acc, lr 0.015] (magenta o), [EM, lr 0.001] (blue x), [EM, lr 0.005], (green x).

Figure 9: Effect of different gradient update rules on convergence rate and log likelihood in the two relation / four visible variable parity problem.

such that

$$\begin{aligned} P(\mathbf{S}) &= \frac{e^{-H(\mathbf{S})}}{Z} \\ H(\mathbf{S}) &= -\frac{\beta}{2}\sum_{i,j} w_{ij} S_i S_j \\ Z &= \sum_{\mathbf{S}'} e^{-H(\mathbf{S}')} \end{aligned}$$

where $w_{ij} = w_{ji}$ and $w_{ii} = 0$.

Traditionally, configurations are sampled from this probability distribution model using Gibbs sampling [12, 39]. When Gibbs sampling is performed according to

$$\begin{aligned} p(S_k^{t+1} = 1 | S_{i\neq k}^t) &= \frac{p(S_k = 1, S_{i\neq k}^t)}{p(S_k = 1, S_{i\neq k}^t) + p(S_k = -1, S_{i\neq k}^t)} \\ &= \frac{e^{\frac{\beta}{2}\sum_{ij} w_{ij}(S_i^t|S_k=1)(S_j^t|S_k=1)}}{e^{\frac{\beta}{2}\sum_{ij} w_{ij}(S_i^t|S_k=1)(S_j^t|S_k=1)} + e^{\frac{\beta}{2}\sum_{ij} w_{ij}(S_i^t|S_k=-1)(S_j^t|S_k=-1)}} \\ &= \frac{e^{\frac{\beta}{2}\left[2\cdot\sum_j w_{kj}S_j^t\right]} \cdot e^{\frac{\beta}{2}\sum_{ij\neq k} w_{ij}S_i^t S_j^t}}{e^{\frac{\beta}{2}\left[2\cdot\sum_j w_{kj}S_j^t\right]} \cdot e^{\frac{\beta}{2}\sum_{ij\neq k} w_{ij}S_i^t S_j^t} + e^{\frac{\beta}{2}\left[-2\cdot\sum_j w_{kj}S_j^t\right]} \cdot e^{\frac{\beta}{2}\sum_{ij\neq k} w_{ij}S_i^t S_j^t}} \\ &= \frac{1}{1 + e^{-\beta\left[2\sum_j w_{kj}S_j^t\right]}} \end{aligned}$$

where $\mathbf{S}^t$ is the previous configuration of the variables and we write $S_{i\neq k}^t$ as a shorthand for $\mathbf{S}^t \backslash k$, then the probability distribution converges to $P(\mathbf{S}) = \frac{e^{-H(\mathbf{S})}}{Z}$, as desired.

In undirected graphical models, the variables are merely arguments to the relations and thus the identity of their values is arbitrary. We could just as easily use $X_{a'} \in \{$"apple", "banana", "cantaloupe"$\}$, so long as $f_{a'}(X_{a'})$ is also defined in terms of these unusual values. In contrast, the standard formulation of Boltzmann machines uses the values of the variables directly in definition of the probability distribution. Thus, to reformulate a Boltzmann

machine as a graphical model, we must select the entries of the relations to account for this difference. We can construct a corresponding undirected graphical model by choosing:

$$\left.\begin{array}{rcl} f_{ij}(-1,-1) = f_{ij}(1,1) & = & e^{\beta w_{ij}} \\ f_{ij}(-1,1) = f_{ij}(1,-1) & = & e^{-\beta w_{ij}} \end{array}\right\} \text{ if } i \neq j$$

$$\left.\begin{array}{rcl} f_{ij}(-1,-1) = f_{ij}(1,1) & = & 1 \\ f_{ij}(-1,1) = f_{ij}(1,-1) & = & 1 \end{array}\right\} \text{ if } i = j$$

where f_{ij} is the relation between variables i and j. Note that there is only one factor between i and j, so the proper notation would probably be $f_{\{i,j\}}$. With these assignments, the probability distribution of the associated undirected graphical model is

$$\begin{aligned} p(\mathbf{X}) &= \frac{\prod_{\{ij\}} f_{ij}(X_{ij})}{Z} \\ &= \frac{e^{\beta \sum_{\{ij\}} w_{ij}(1-2\cdot(X_i \oplus X_j))}}{Z} \\ &= \frac{e^{\frac{\beta}{2} \sum_{ij} w_{ij} S_i S_j}}{Z} \end{aligned}$$

This is exactly the same as the original Boltzmann machine. The Boltzmann dynamics can be recaptured by performing Gibbs sampling on this undirected graphical model. However, if approximate results are sufficient, statistical inference can be performed much more quickly using belief propagation.

4.1 Mean-field Boltzmann machines

Another well-known fast, approximate method for statistical inference on Boltzmann machines is the deterministic/mean field Boltzmann machine update rule [12, 41], defined by

$$\begin{aligned} m_i &= \tanh\left(\beta \sum_j w_{ij} m_j\right) \\ &= \frac{e^{\beta \sum_j w_{ij} m_j} - e^{-\beta \sum_j w_{ij} m_j}}{e^{\beta \sum_j w_{ij} m_j} + e^{-\beta \sum_j w_{ij} m_j}} \end{aligned}$$

This update rule is based on the approximation $m_i = \langle S_i \rangle$. It follows that

$$\begin{aligned} \langle S_i \rangle &= p(S_i = 1) \cdot 1 + p(S_i = -1) \cdot (-1) \\ &= \frac{1}{1 + e^{-\beta\left[2 \sum_j w_{ij} S_j\right]}} - \frac{1}{1 + e^{\beta\left[2 \sum_j w_{ij} S_j\right]}} \\ &= \frac{e^{\beta\left[\sum_j w_{ij} S_j\right]} - e^{-\beta\left[\sum_j w_{ij} S_j\right]}}{e^{\beta\left[\sum_j w_{ij} S_j\right]} + e^{-\beta\left[\sum_j w_{ij} S_j\right]}} \\ &= \tanh\left(\beta\left[\sum_j w_{ij} S_j\right]\right) \end{aligned}$$

The mean field approximation is obtained by replacing S_i with $\langle S_i \rangle$ on the right.

$$\langle S_i \rangle \approx \tanh\left(\beta \sum_j w_{ij} \langle S_j \rangle\right)$$

Rather than performing Gibbs sampling on the original Boltzmann machine, we can define new variables $m_i = \langle S_i \rangle$ and perform deterministic updates according to the rule above:

$$m_i \Leftarrow \tanh\left(\beta \sum_j w_{ij} m_j\right)$$

The m_i will not reach their final values after a single round of this update; rather, it must be iterated until convergence is achieved. Like belief propagation, this scheme features a local representation for each variable and updates based upon the local graph structure.

Since each variable can only take on two possible values, we can recover $p(X_i = -1)$ and $p(X_i = 1)$ from m_i.

$$m_i = p(X_i = 1) \cdot 1 + p(X_i = -1) \cdot -1 = p(X_i = 1) - (1 - p(X_i = -1)) = 2 \cdot p(X_i = 1) - 1$$

Plugging this into the mean field Boltzmann machine update rule and noting that $\frac{e^x - e^{-x}}{e^x + e^{-x}} = 2 \cdot \frac{1}{1+e^{-2x}} - 1$, we obtain

$$\begin{aligned}
2 \cdot p(X_i = 1) - 1 &\Leftarrow \frac{e^{\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]} - e^{-\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]}}{e^{\beta\left[\sum_j w_{ij}(2 \cdot p(X_j=1) - 1)\right]} + e^{-\beta\left[\sum_j w_{ij}(2 \cdot p(X_j=1) - 1)\right]}} \\
&= 2 \cdot \frac{1}{1 + e^{-2\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]}} - 1 \\
p(X_i = 1) &\Leftarrow \frac{1}{1 + e^{-2\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]}} \\
&= \frac{e^{\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]}}{e^{\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]} + e^{-\beta\left[\sum_j w_{ij}(1 \cdot p(X_j=1) + (-1) \cdot p(X_j=-1))\right]}} \\
&= \frac{\prod_j \left(e^{\beta w_{ij}}\right)^{p(X_j=1)} \cdot \left(e^{-\beta w_{ij}}\right)^{p(X_j=-1)}}{\prod_j \left(e^{\beta w_{ij}}\right)^{p(X_j=1)} \cdot \left(e^{-\beta w_{ij}}\right)^{p(X_j=-1)} + \prod_j \left(e^{-\beta w_{ij}}\right)^{p(X_j=1)} \cdot \left(e^{\beta w_{ij}}\right)^{p(X_j=-1)}} \\
&= \frac{\prod_j f_{ij}(1,1)^{p(X_j=1)} \cdot f_{ij}(1,-1)^{p(X_j=-1)}}{\prod_j f_{ij}(1,1)^{p(X_j=1)} \cdot f_{ij}(1,-1)^{p(X_j=-1)} + \prod_j f_{ij}(-1,1)^{p(X_j=1)} \cdot f_{ij}(-1,-1)^{p(X_j=-1)}} \\
&= \sum_{\mathbf{X}|X_i=1} \frac{\prod_j f_{ij}(1,1)^{p(X_j=1)} \cdot f_{ij}(1,-1)^{p(X_j=-1)} \cdot \prod_{j,k \neq i} f_{jk}(X_j, X_k)}{Z'} \\
&= \frac{\sum_{\mathbf{X}|X_i=1} \prod_j f_{ij}(1,1)^{p(X_j=1)} \cdot f_{ij}(1,-1)^{p(X_j=-1)} \cdot \prod_{j,k \neq i} f_{jk}(X_j, X_k)}{Z'}
\end{aligned}$$

where

$$Z' = \sum_{\mathbf{X}|X_i=1} \left(\prod_j f_{ij}(1,1)^{p(X_j=1)} \cdot f_{ij}(1,-1)^{p(X_j=-1)} + \prod_j f_{ij}(-1,1)^{p(X_j=1)} \cdot f_{ij}(-1,-1)^{p(X_j=-1)} \right) \cdot \prod_{j,k \neq i} f_{jk}(X_j, X_k)$$

This is like a weighted geometric mean. Since there are only two possible values for each variable, $p(X_i = -1) = 1 - p(X_i = 1)$ and all of the marginal variables are specified. Note that $w_{ij} = 0$ if variables X_i and X_j are not connected, so $f(X_i, X_j) = 1$ regardless of the values of X_i and X_j. Deterministic / mean field Boltzmann machines thus bear a striking resemblance to belief propagation, in that the marginal probabilities of the variables are calculated by means of an iterated local update.

We can also write the update as

$$p(X_i = 1) \Leftarrow \frac{\prod_j e^{\beta[w_{ij}(X_i=1)\langle X_j \rangle]}}{\prod_j e^{\beta[w_{ij}(X_i=1)\langle X_j \rangle]} + \prod_j e^{\beta[w_{ij}(X_i=-1)\langle X_j \rangle]}}$$

In this form, the main difference between deterministic / mean field Boltzmann machines and belief propagation lies in where the averaging is performed. Whereas deterministic / mean field Boltzmann machines replace X_i with its average $\langle X_i \rangle = m_i$, belief propagation calculates the full update rule for both $X_i = -1$ and $X_i = 1$ and then uses the linear combination weighted by $p(X_i = \pm 1)$, as we shall see below. Belief propagation also differs from deterministic / mean field Boltzmann machines in the order in which updates are performed. A single deterministic / mean field Boltzmann machine update for a variable i, which is directly connected to a set of variables $N(i)$, most closely corresponds to the set of belief-propagation updates from all variables $j \in N(i)$ to i. Since each of these

messages passes through a single relation of degree two, they have magnitude

$$\begin{aligned} m_{j\to i}(X_i=1) &= \left(\prod_{k\in N(j)\setminus i} m_{k\to j}(X_j=1)\right)e^{\beta w_{ij}} + \left(\prod_{k\in N(j)\setminus i} m_{k\to j}(X_j=-1)\right)e^{-\beta w_{ij}} \\ m_{j\to i}(X_i=-1) &= \left(\prod_{k\in N(j)\setminus i} m_{k\to j}(X_j=-1)\right)e^{\beta w_{ij}} + \left(\prod_{k\in N(j)\setminus i} m_{k\to j}(X_j=1)\right)e^{-\beta w_{ij}} \end{aligned}$$

where i, j, and k represent variables rather than relations. Since

$$\begin{aligned} m^{old}_{i\to j}(X_j=1) &= \left(\frac{p(X_i=1)}{m_{j\to i}(X_i=1)}\cdot e^{\beta w_{ij}} + \frac{p(X_i=0)}{m_{j\to i}(X_i=0)}\cdot e^{-\beta w_{ij}}\right) \\ p(X_j=1) &\propto \prod_{k\in N(j)} m_{k\to j}(X_j=1) \\ &\propto \left(\frac{p(X_i=1)}{m_{j\to i}(X_i=1)}\cdot e^{\beta w_{ij}} + \frac{p(X_i=0)}{m_{j\to i}(X_i=0)}\cdot e^{-\beta w_{ij}}\right)\cdot \prod_{k\in N(j)\setminus i} m_{k\to j}(X_j=1) \\ m^{new}_{j\to i}(X_i=1) &= \frac{1}{\frac{p(X_i=1)}{m^{old}_{j\to i}(X_i=1)}\cdot e^{\beta w_{ij}} + \frac{p(X_i=0)}{m^{old}_{j\to i}(X_i=0)}\cdot e^{-\beta w_{ij}}} p(X_j=1)e^{\beta w_{ij}} + \\ &\quad \frac{1}{\frac{p(X_i=1)}{m^{old}_{j\to i}(X_i=1)}\cdot e^{-\beta w_{ij}} + \frac{p(X_i=0)}{m^{old}_{j\to i}(X_i=0)}\cdot e^{\beta w_{ij}}} p(X_j=-1)e^{-\beta w_{ij}} \\ &= \alpha_1 p(X_j=1)e^{\beta w_{ij}} + \alpha_2 p(X_j=-1)e^{-\beta w_{ij}} \\ p(X_i=1) &= \frac{\prod_{j\in N(i)} m^{new}_{j\to i}(X_i=1)}{Z} \\ &\approx \frac{\prod_{j\in N(i)} \left\langle e^{\beta w_{ij} X_j}\right\rangle}{Z} \end{aligned}$$

where equality holds if $w_{ij}=0$ or $\frac{p(X_i=1)}{m^{old}_{j\to i}(X_i=1)} = \frac{p(X_i=0)}{m^{old}_{j\to i}(X_i=0)}$

4.2 Training Boltzmann machines

We can use expectation maximization to derive a novel algorithm for training Boltzmann machines.

$$\begin{aligned} 0 &= \frac{\partial}{\partial w_{ij}} E[\log p(\mathbf{X}) \mid \mathbf{Y}, \theta^t] \\ &= \frac{\partial}{\partial w_{ij}}\left(-n\log\left(\sum_X \prod_a f_a(X_a)\right) + \sum_{a,X_a}\left(\sum_i p\left(X^i_a|Y^i,\theta^t\right)\right)\cdot \log f_a(X_a)\right) \\ &= -n\beta\frac{\sum_{X|(X_i\oplus X_j=0)}\prod_a f_a(X_a) - \sum_{X|(X_i\oplus X_j=1)}\prod_a f_a(X_a)}{\sum_X \prod_a f_a(X_a)} \\ &\quad +\beta\sum_k \big(p(X^k_i=0,X^k_j=0|Y^k,\theta^t) + p(X^k_i=1,X^k_j=1|Y^k,\theta^t) - p(X^k_i=0,X^k_j=1|Y^k,\theta^t) \\ &\quad - p(X^k_i=1,X^k_j=0|Y^k,\theta^t)\big) \end{aligned}$$

$$\frac{e^{\beta w_{ij}}\sum_{X|(X_i\oplus X_j=0)}\prod_{a\neq\alpha} f_a(X_a) - e^{-\beta w_{ij}}\sum_{X|(X_i\oplus X_j=1)}\prod_{a\neq\alpha} f_a(X_a)}{e^{\beta w_{ij}}\sum_{X|(X_i\oplus X_j=0)}\prod_{a\neq\alpha} f_a(X_a) + e^{-\beta w_{ij}}\sum_{X|(X_i\oplus X_j=1)}\prod_{a\neq\alpha} f_a(X_a)} = \frac{1}{n}\sum_{k,X'_i,X'_j} p(X^k_i=X'_i,X^k_j=X'_j|Y^k,\theta^t)\cdot\left(1-2\cdot(X'_i\oplus X'_j)\right)$$

We make the assumption that Z is constant and the substitutions

$$\begin{aligned}
\log(W) &= \beta w_{ij} \\
P &= \frac{1}{n} \sum_{k,X'_i,X'_j} p(X_i^k = X'_i, X_j^k = X'_j | Y^k, \theta^t) \cdot \left(1 - 2 \cdot (X'_i \oplus X'_j)\right) \\
Z &= \sum_X \prod_a f_a(X_a) \\
N_x &= \sum_{X|(X_i \oplus X_j = x)} \prod_{a \neq \alpha} f_a(X_a)
\end{aligned}$$

we find

$$\begin{aligned}
\frac{WN_0 - \frac{1}{W}N_1}{Z} &= P \\
W &= \frac{PZ \pm \sqrt{(PZ)^2 + 4N_0N_1}}{2N_0} \\
w_{ij} &= \frac{1}{\beta} \log\left(\frac{PZ \pm \sqrt{(PZ)^2 + 4N_0N_1}}{2N_0}\right)
\end{aligned}$$

However, we can solve the full expression for w_{ij} just as easily:

$$\begin{aligned}
\frac{WN_0 - \frac{1}{W}N_1}{WN_0 + \frac{1}{W}N_1} &= P \\
W &= \sqrt{\frac{N_1(1+P)}{N_0(1-P)}} \\
w_{ij} &= \frac{1}{\beta} \log\left(\sqrt{\frac{N_1(1+P)}{N_0(1-P)}}\right)
\end{aligned}$$

5 Connection to Biology

5.1 Preface

As any practicing scientist knows, scientific progress does not strictly follow the Kuhnian model of testing and disproving theories. Modern neuroscience experiments contain countless factors which are all but impossible to control and which may confound results in complicated ways. For instance, it is likely that almost all electrophysiological investigations contain a substantial bias in the selection of neurons from which to record. Electrophysiology is often performed without visual guidance, so the neurons selected are those which produce the most obvious signal in the electrode. If signal strength varies with neuron size or type (e.g., pyramidal neurons might produce a stronger electrode signal than inhibitory interneurons), then there will be a similar bias in the frequency of recordings from such cells. Additionally, most experiments probing the firing properties of single cells are performed either in slices or anesthetized animals. Both of these preparations have significant and poorly-understood affects on network activity in the brain [44, 47]. Neurons can operate in many different regimes, governed by neuromodulators, firing rate and burst tendencies, and statistical properties of the afferent signals. While a particular cellular phenomenon may be extremely robust in one regime, its importance for overall brain function cannot be assessed without knowledge of the context in which the neuron operates.

No known recording technique can record from enough cells at sufficient spatial and temporal resolution to definitively answer basic questions about the operations of networks of neurons. At best, extracellular electrophysiology with dozens of electrodes can record from a few hundred cells, whereas a single cortical column can contain over 200 cells [34]. fMRI can record from substantial regions of the brain, but it can neither isolate single cells nor single spikes. Calcium- and voltage-sensitive dyes are promising, but their scan rates are finite and their ability to pierce through many layers of cortex is limited [40].

Given that our experimental techniques are limited in their power, it is necessary for experiments to be motivated by clear hypotheses. While any grand theory of brain function will invariably be incorrect in its details (and often its basic premise), overarching theories can provide a useful basis for experimental work. Occasionally, such theories may even contain a kernel of truth. Marr's theory of the cerebellum and archaecortex are still heavily cited [28, 29]; Hubel and Wiesel's theories of visual receptive fields continue to steer current thinking [14].

It is in this light that we present an interpretation of the cortex as an undirected graphical model. Many of the experimental findings on which this model is based remain controversial, and some may well be invalidated by future experiments. However, we hope it will contain enough surprising correspondences with experimental findings to inspire future experimental and theoretical work.

5.2 Model definition

We propose that the primary function of the cortex is to calculate the marginal probabilities of variables and relations in an undirected graphical model. The cortex accomplishes this task by implementing belief propagation on the factor graph corresponding to the undirected graphical model. Specifically, the cortex computes the outgoing belief propagation messages from each relation by calculating the sum of the products of the belief propagation messages propagating into the relation.

The belief propagation message from relation a_i to variable X_j is

$$m_{a_i \to X_j} \Longleftarrow \sum_{X_{a_i} \setminus X_j} f_{a_i}(X_{a_i}) \prod_{X_k \in N(a_i) \setminus X_j} n_{X_j \to a_i}(X_k)$$

where $f_{a_i}(X_{a_i})$ are the parameters of relation a_i and $N(a_i)$ is the set of variables connected to (i.e., the neighbors of) relation a_i [23, 55]. This update rule is known as the sum-product algorithm. We restrict our consideration to factor graphs where variables only connect pairs of relations. Belief propagation messages pass unchanged through variables which are only connected to two relations. The belief propagation message from variable X_j to relation a_k is thus

$$n_{X_j \to a_k} \Longleftarrow m_{a_i \to X_j}$$

where variable X_j connects relations a_i and a_k.

We propose that clusters of neurons in the brain calculate the belief propagation messages for a single relation in a factor graph. A schematic of the suggested neural structure is shown in Figure 10. Each pyramidal cell within layer 2/3 of each cluster calculates the outgoing belief propagation message corresponding to a single value of one variable to which the relation is connected (a variable/value combination). In Figure 10, each variable (represented by the dotted circles) can take on one of two values (represented by different colors), although in general each variable may take on more values. Multiple neurons represent each variable/value combination and together constitute a population code. Thus, in Figure 10, each schematic neuron represents a population of neurons in the brain.

Axons transmit information in only one direction, whereas in factor graphs information propagates across the graph edges in both directions. In Figure 10, each variable/value combination is represented by two populations of axons. Solid lines carry signals traveling to the neurons implementing a relation (e.g., $m_{X_j \to a_k}$), while dotted lines carry signals traveling away from neurons implementing a relation (e.g., $n_{a_i \to X_j}$). Since belief propagation messages pass unchanged through degree-two variables, the variables need not be explicitly represented by any physical structure in the brain other than the axonal tracts which recurrently connect nearby cortical clusters. The axon lines change from dashed to solid in the circles denoting the variables of the factor graph to indicate that the signal they carry has conceptually changed from the belief propagation message $m_{a_i \to X_j}$ (from the original relation to the variable) to the belief propagation message $n_{X_j \to a_k}$ (from the variable to the next relation). However, these lines still represent the same physical axons carrying the same signals.

The firing rate (or potentially the firing probability within particular time windows) of each neuron corresponds to the belief propagation message of the variable/value combination with which it is associated. Each individual layer 2/3 pyramidal cell implements a sum-product rule over its inputs. Small sections of dendritic branches receive projections from multiple sources corresponding to all of the variables connected to the cortical cluster's relation, other than the variable to which the current neuron's axon projects. Each such section should receive only a single (or nearby) value(s) for each incoming variable. The belief propagation messages for these variable/value combinations, encoded as firing rates or probabilities, are multiplied together within the dendritic branches via a mechanism described below and scaled according to the parameters of the current relation. These products are then summed at the soma, transformed into a firing rate (or firing probability) and communicated (potentially after some sort of nonlinear transformation, which may be undone at the other end) to the recipient areas via the neuron's action potentials.

Consider a population of cells C_i corresponding to relation a_i, which is connected in the factor graph through degree-two variables $X_1, X_2, \ldots X_j, \ldots X_n$ to relations $a_1, a_2, \ldots a_j, \ldots a_n$. We will call a population such as C_i a cluster, although it may correspond to anatomical features such as a cortical column or a daisy patch (REFERENCE?). Note that $\prod_{j=1}^{n} X_j = X_{a_i}$. Cluster C_i contains $\sum_{j=1}^{n} |X_j|$ distinct populations of neurons, where $|X_j|$ is the number of values assumed by X_j. Call the population of cells in cluster C_i corresponding to value v of variable X_j $P^i_{j,v}$. Ideally, for every cell in $P^i_{j,v}$, each dendritic branch should receive one projection (corresponding to a single value) from each of the variables other than j. Thus, each branch should receive $n-1$ synapses (where n is the number of variables to which relation a_i is connected), and there are $\prod_{1 \leq k \leq n, k \neq j} |X_k|$ unique branches. Each cell in population $P^i_{j,v}$ should project exclusively to cluster C_j, where it should terminate on every cell except those projecting back to cluster C_i.

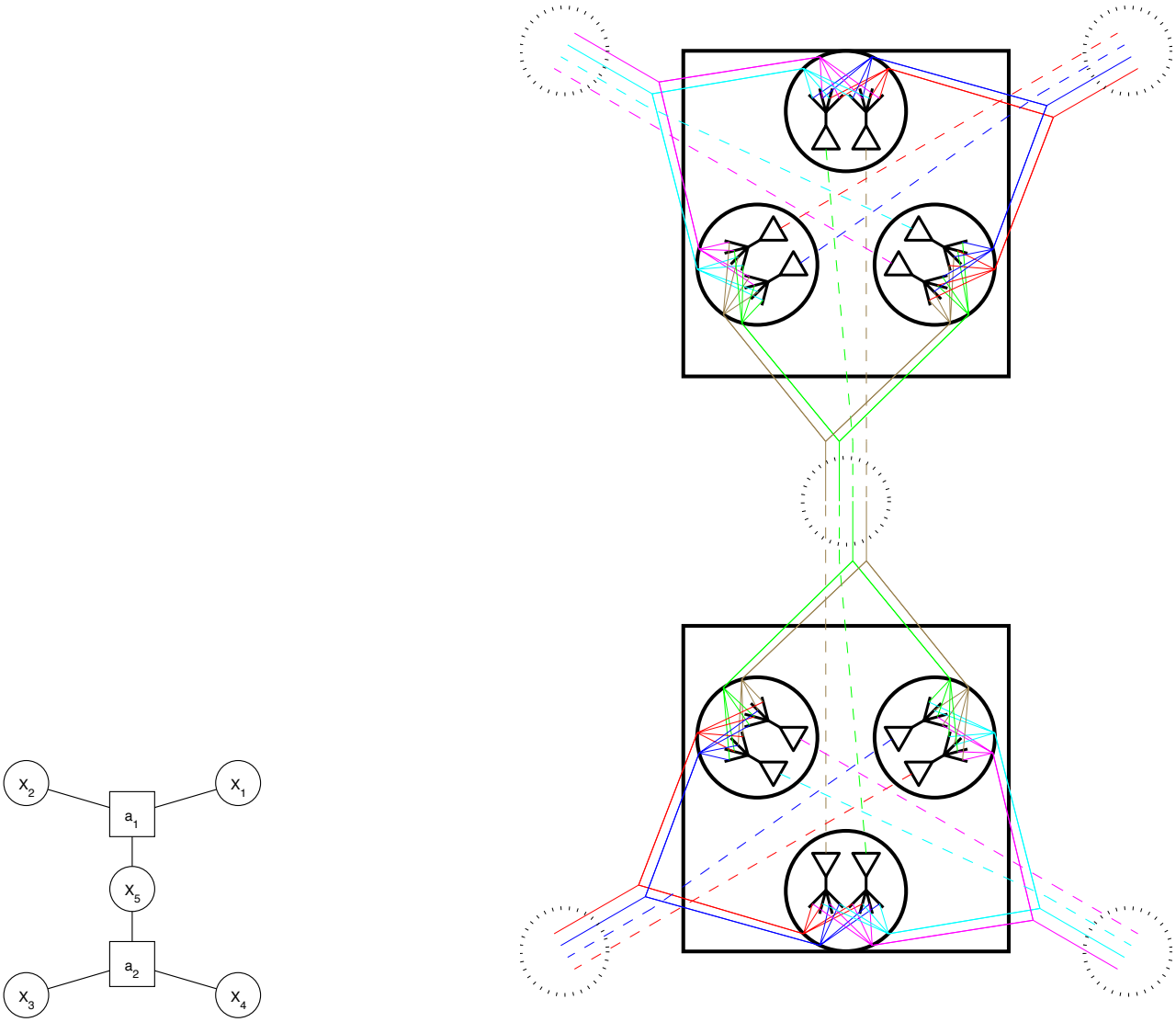


Figure 10: A small factor graph and its neural implementation

5.3 Dendritic multiplication

Many mechanisms have been proposed by which multiplication can be performed in single neurons (see [21, 22, 33] for review). However, few proposed mechanisms perform exact multiplication. Koch's dirty multiplication with shunt

inhibition computes an approximation to multiplication, which becomes increasingly inaccurate for large values of the excitatory synaptic conductance. Mel's sigma-pi neurons are based upon a mechanism similar to that described below, but their dendritic nonlinearity was never intended to be exact. Summing logarithms requires first calculating the logarithm of inputs, a nontrivial operation using neuronal biophysics. In contrast, multiplication of firing rates via coincidence detection can be relatively precise [6, 46].

We propose that multiplication is performed via coincidence detection in distal dendritic branches [32]. Numerous studies have found that simultaneous activation of multiple synapses within a single small dendritic branch can induce somatic depolarization nonlinear in the magnitude of the synaptic activation [4, 11, 26, 42, 45, 48]. The nature and magnitude of this nonlinearity depend upon the details of the experimental protocol. Dendrites contain a variety of channels that are sensitive to the local potential, including voltage-gated Na^+ channels, voltage-gated Ca^{2+} channels, and NMDA channels [11]. Correspondingly, dendritic spikes have been reported that are driven primarily by Na^+ in radial oblique dendrites of CA1 [26], combined Na^+ and NMDA channels in the basal dendrites of CA1 [4], and NMDA-mediated Ca^{2+} in basal dendrites of layer 5 pyramidal cells [45].

Although the density of channels such as the voltage gated Na^+ channel can be as large in the distal dendrites as it is in the soma, it still falls short of the high concentration of voltage-gated channels at the first node of Ranvier in the axon, so the local threshold for dendritic spike initiation is larger than the axosomatic spike threshold. However, in layer 5 cortical pyramidal neurons, the local amplitude of apical EPSPs increases exponentially with the distance of the synapse from the some, while somatic depolarization due to a dendritic EPSP decreases exponentially with distance from the soma [52]. Thus, dendritic spikes can occur at levels of synaptic activity that fall far short of directly inducing an axosomatic spike. Williams and Stuart [52] estimate that between 4 and 20 coincident synaptic inputs are sufficient to trigger a dendritic spike in the apical dendrites of CA1 pyramids. Losonczy and Magee [26] concur that about 20 synaptic inputs within 6 ms are necessary to generate a dendritic spike in the radial oblique dendrites of CA1. Fewer inputs may be sufficient in thin, distal dendrites at the tips of the basal and apical tufts, but despite comprising the vast majority of most dendritic trees, these regions contain only a small minority of the neuron's spines [24].

Whereas dendritic spikes originating in the apical dendrites of layer 5 pyramidal cells can independently trigger multiple axosomatic spikes [52, 53], dendritic spikes generated in the basal dendrites of layer 5 pyramidal cells [45] and the radial oblique dendrites of CA1 cells [26] generally result in somatic depolarizations no more than twice as large as those resulting from the linear combination of the inputs. The somatic depolarization is limited after basal dendritic spikes because the spike is local and does not propagate beyond the branch in which it was generated [4, 48]. In these cases, the additional dendritic depolarization due to the spike can propagate to the soma via purely passive mechanisms, but it may also be subject to nonlinear boosting (which falls short of inducing a full spike) along the way.

This interpretation of the neuron as a two-layer network bears a strong similarity to Poirazi et al.'s model of the neuron as a two-layer integrate-and-fire network [37]. However, in this case we take advantage of the small capacitance and thus small time constant of the dendritic branches to perform a different operation than in the soma, which has a large capacitance and associated time constant [6, 33].

The strength of these synapses and the local nonlinear amplification correspond to the weight of the entry for the particular variable/value combinations synapsing on that section of dendrite. Note that the patchy connections formed between local cortical regions arise from pyramidal cells and thus are almost certainly excitatory [8]. This is compatible with the sum-product rule of an undirected graphical model, but is contradictory to the usual surround-inhibition conception of visual cortex processing.

5.4 Anatomy

This model makes a few well-defined predictions about cortical anatomy:

- Cortex must be organized into local units, of which only a fraction are active in response to any stimulus
- Reciprocal connections must exit between particular pairs of units. There should be a clear distinction between connected and unconnected pairs of units. Synaptic connections within a unit may be diffuse, but there should not be a diffuse projection of synapses between unconnected units.
- Individual neurons within a unit must not project to units from which they receive input. Specifically, although units should always be recursively linked, pairs of neurons in different units should never be recursively linked.
- Connections within units may be dense, but they must not distribute activity in such a way that the entire unit is activated in an all-or-none manner.

Cortical anatomy seems to be compatible with these predictions. Electrophysiology [34] and optical imaging with calcium-sensitive dyes [40] support the conclusion that the cortex is divided into functional columns on the order of 500μm in diameter, in which cells respond to the same type of inputs. We propose that each such column corresponds to the cluster of cells which implements a single relation in the factor graph. When tracers are injected into a small region of the cortex, both efferent and afferent projections form patches with a diameter comparable to that of the functional columns [3, 49]. The labeled anterogradely labeled axons and retrogradely labeled cell bodies overlap, implying that these patchy connections are reciprocal. We suggest that these patches correspond to the connections between relations described in Subsection 5.2.

When a neuron projects to distal columns, it should connect relatively indiscriminately within the column, since all neurons within the column need its information. Douglas and Martin [8] note that the diameter of axonal patches matches the diameter of the dendritic arbor of the neurons at the patch site. While they suggest that the dendrites are uniformly distributed around the soma, such that most neurons lie half in and half out of neighboring patches, perhaps the dendritic arbors are actually aligned with the patches In this case, the number of local dendritic sites at which one input could be combined with others (each constituting one cell in the relation table) would be maximized.

In general pyramidal cells receive afferents from a variety of different areas [3] (assuming that the dendritic trees connect to nearby axons indiscriminately) and axons project to only a few different clusters [10], so this scheme is not totally implausible. However, this ideal state of affairs is probably inaccurate in a variety of ways. Each cell may not receive every possible product, and some products may be represented repeatedly. Products may not contain terms from every afferent cluster. Cells may project to clusters from which they receive projections. In particular, cells may project to multiple clusters. The implications of these discrepancies will be the subject of future work.

It has previously been proposed that logical computations, such as $X \wedge \bar{Y}$ (AND-NOT), could be computed within single dendritic branches [21]. Since AND and NOT are universal boolean operations, any boolean function could be computed in the dendritic tree. However, this model requires the precise connection of synapses from particular neurons on a single dendritic branch. In the case of $X \wedge \bar{Y}$, the input corresponding to Y must form an inhibitory synapse between the segment of the dendritic tree implementing X and all other synaptic inputs. Since the computation performed is determined by the spatial ordering of the synapses, they cannot occur at random or random functions would result.

In our model, multiple neurons represent each variable/value combination, so single neurons can fail to fire or die without affecting the overall operation. Any particular neuron need not connect to any other particular neuron, or in conjunction with any other particular synapse. All that is necessary is that the neurons corresponding to each variable/value combination project densely to the destination columns. One would thus expect that individual neurons in a cluster would project to only a fraction of the set of columns to which other neurons in the given cluster project. These destinations of single neurons should cluster: all of the neurons corresponding to a single variable should project to the same destination, which is distinct from the destination of neurons corresponding to different variables.

5.5 Learning

The expectation maximization update rule is very similar to what we observe in the brain. Synapses, corresponding to relation weights, should be strengthened in proportion to the activation/firing rate of the neuron, which represents the probability conditioned on the current input of the represented variable/value combination. However, all synapses of the neuron should not be strengthened; only those corresponding to the present configuration; i.e., the active inputs. This is very similar to the synaptic weight changes observed during spike-timing dependent plasticity [1].

Using the EM update rules, it is not necessary to have separate wake and sleep phases. Instead, the unlearning of previous weights is performed through a multiplicative normalization term, $p(X_a = x)$ In fact, this is the same as the term subtracted in the sleep phase wake-sleep algorithm. Were the sleep phase of the wake-sleep algorithm to occur separately from the wake phase, in the context of our model of the cortex as an undirected graphical model it would require that synaptic weight dynamics completely reverse. During a distinct sleep phase synaptic weights would have to decrease in response to presynaptic activity inducing axosomatic spikes. This is the opposite of observed spike-time dependent plasticity.

In contrast, experimental evidence supports the scaling of synaptic strength by the overall probability of activation [1]. The strength of LTP induced by presynaptic activity followed by a spike is proportional to the initial strength of the synapse [5]. Many experiments have found that reduction of overall activity via TTX (or similar substances) increases synaptic strength, whereas an increase in activity reduces synaptic strength. We will continue to investigate the relationship between learning rules for undirected graphical models and neural synaptic weight changes in future work.

References

[1] Abbot, L. F., and Nelson, S. B. (2000). Synaptic plasticity: Taming the beast. Nature Neuroscience, 3, 1178-1183.

[2] Aji, S. M., and McEliece, R. J. (2000). The generalized distributive law. IEEE Transactions on Information Theory 46(2), 325-343.

[3] Angelucci, A., Levitt, J. B., Walton, E. J. S., Hupe, J-M., Bullier, J., and Lund, J. S. (2002). Circuits for local and global signal integration in primary visual cortex. The Journal of Neuroscience 22(19), 8633-8646.

[4] Ariav, G., Polsky, A., and Schiller, J. (2003). Submillisecond precision of the input-output transformation function mediated by fast sodium dendritic spikes in basal dendrites of CA1 pyramidal neurons. Journal of Neuroscience 23(21), 7750-7758.

[5] Bi, G-Q, and Poo, M-M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of Neuroscience 18(24), 10464-10472.

[6] Bugmann, G. (1991). Summation and multiplication: Two distinct operation domains of leaky integrate-and-fire neurons. Network 2, 489-509.

[7] Dayan, P. and Abbott, L. F. (2001). Theoretical neuroscience: Computational and mathematical modeling of neural systems. MIT Press

[8] Douglas, R. J., and Martin, K. A. C. (2004). Neuronal circuits of the neocortex. Annual Review of Neuroscience 27, 419-451.

[9] Freeman, W., Pasztor, E. C., and Carmichael, O. T. (2000). Learning low-level vision. International Journal of Computer Vision 40(1), 25-47.

[10] Gilbert, C. D., and Wiesel, T. N. (1983). Clustered intrinsic connections in cat visual cortex. The Journal of Neuroscience 3(5), 1116-1133.

[11] Hausser, M., Spruston, N., and Stuart, G. (2000). Diversity and dynamics of dendritic signaling. Science 290, 739-744.

[12] Hertz, J., Krogh, A., and Palmer, R. G. (1991). Introduction to the theory of neural computation. FINISH

[13] Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations. (pp 282-317) Cambridge: MIT Press

[14] Hubel, D. H., and Wiesel, T. N. (1962). Receptive fields, binocular interaction, and functional architecture in cat's visual cortex. Journal of Physiology (London) 160(1), 106-???.

[15] Kandel, E. R., Schwatrz, J. H., and Jessell, T. M. (2000). Principles of Neural Science, 4th ed. McGraw-Hill, New York.

[16] Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. Computational Statistics and Data Analysis (19), 191-201.

[17] Hines, M. L. and Carnevale, N. T. (2002). The NEURON simulation environment. In The Handbook of Brain Theory and Neural Networks, 2nd Ed. (M. A. Arbib, ed.), Cambridge, MA: The MIT Press.

[18] Jordan, M. I. (2004). Graphical models. Statistical Science 19(1), 140-155.

[19] Kersten, D., Mamassian, P., and Yuille, A. (2004). Object perception as Bayesian inference. Annual Review of Psychology 55, 271-304.

[20] Knill, D. C. and Pouget, A. (2004). The Bayesian brain: The role of uncertainty in neural coding and computation. TRENDS in Neurosciences 24(12), 712-719.

[21] Koch, C. (1999). Biophysics of computation: Information processing in single neurons. New York, Oxford University Press.

[22] Koch, C. and Segev, I. (2000). The role of single neurons in information processing. Nature Neuroscience 3, 1171-1177.

[23] Kschischang, F. R. (2001) Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498-519.

[24] Larkman, A. U. (1991) Dendritic morphology of pyramidal neurones of the visual cortex of teh rat: III. Spine distributions. The Journal of Comparative Neurology 306, 332-343.

[25] Lee, T. S. and Mumford, D. (2003) Hierarchical Bayesian inference in the visual cortex. Journal of the Optical Society of America 20(7), 1434-1448.

[26] Losonczy A., and Magee, J. C. (2006). Integrative properties of radial oblique dendrites in hippocampal CA1 pyramidal neurons. Neuron 50, 291-307.

[27] MacKay, D. J. C. (2003). Information Theory, Inference, and Learning Algorithms. Cambridge: Cambridge University Press.

[28] Marr, D. (1969). A theory of cerebellar cortex. Journal of Physiology (London) 202(2), 437-???.

[29] Marr, D. (1971). Simply memory; Theory for archicortex. Philosophical Transactions of the Royal Society of London Series B - Biological Sciences 262(841), 23-???.

[30] Marr, D. (1982). Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. New York: W. H. Freeman and company.

[31] McLachlan, G. J., and Krishnan, T. (1997). The EM algorithm and extensions.

[32] Mel, B. W. (1993). Synaptic integration in an excitable dendritic tree. Journal of Neurophysiology 70(3), 1086-1101.

[33] Mel, B. W. (1994). Information processing in dendritic trees. Neural Computation 6, 1031-1085.

[34] Mountcastle, V. B. (1997). The columnar organization of the neocortex. Brain 120, 701-722.

[35] Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. Uncertainty in Artificial Intelligence 1999.

[36] Murray, I., and Ghahramani, Z. (2004). Bayesian learning in undirected graphical models: Approximate MCMC algorithms. Uncertainty in Artificial Intelligence 2004.

[37] Poirazi, P., Brannon, T. M., and Mel B. W. (2003). Pyramidal neuron as two-layer neural network. Neuron 37, 989-999.

[38] Press, W. H., Teukolsky, S. A., Vetterline, W. T., and Flannery, B. P. (1992). Numerical recipes in C: The art of scientific computing (2nd Ed).

[39] Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. University of Toronto Department of Computer Science Technical Report CRG-TR-93-1.

[40] Ohki, K. Chung, S. Ch'ng, Y. H., Kara, P., and Reid, R. C. (2005). Functional imaging with cellular resolution reveals precise microarchitecture in visual cortex. Nature 433, 597-603.

[41] Peterson, C., and Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. Complex Systems 1, 995-1019.

[42] Polsky, A., Mel, B. W., and Schiller, J. (2004). Computational subunits in thin dendrites of pyramidal cells. Nature Neuroscience 7(6), 621-627.

[43] Rao, R. P. N. (2004). Bayesian computation in recurrent neural circuits. Neural Computation 16, 1-38.

[44] Rudolph, U. and Antkowiak, B. (2004). Molecular and neuronal substrates for general anaesthetics. Nature Reviews Neuroscience 5, 709-720.

[45] Schiller, J., Major, G., Koester, H. J., and Schiller, Y. (2000). NMDA spikes in basal dendrites of cortical pyramidal neurons. Nature 404, 285-289.

[46] Srinivasan, M. V., and Bernard, G. D. (1976). A proposed mechanism for multiplication of neural signals. Biological Cybernetics 21, 227-236.

[47] Steriade, M. (2004). Neocortical cell classes are flexible entities. Nature Reviews Neurosciences 5, 121-134.

[48] Wei, D-S., Mei, Y-A., Bagal, A., Kao, J. P. Y., Thompson, S. M., Tang, C-M. (2001). Compartmentalized and binary behavior of terminal dendrites in hippocampal pyramidal neurons. Science 293, 2272-2275.

[49] Tanigawa, H., Wang, Q., and Fujita, I. (2005). Organization of horizontal axons in the inferior temporal cortex and primary visual cortex of the Macaque monkey. Cerebral Cortex 15, 1887-1899.

[50] Teh, Y. W., and Welling, M. (2001). The unified propagation and scaling algorithm. Neural Information Processing Systems 2001.

[51] Wiegerinck, W., and Heskes, T. (2002). IPF for discrete chain factor graphs. Uncertainty in Artificial Intelligence, Proceedings of the Eighteenth Conference, 560-567.

[52] Williams, S. R., and Stuart, G. J. (2002). Dependence of EPSP efficacy on synapse location in neocartical pyramidal neurons. Science 295, 1907-1910.

[53] Williams, S. R. (2004). Spatial compartmentalization and functional impact of conductance in pyramidal neurons. Nature Neuroscience 7(9), 961-967.

[54] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2002). Understanding belief propagation and its generalizations. Interantional Joint Conference on Artificial Intelligence

[55] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2005). Constructing free energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory 51, 2282-2313.