Graphical models: parameter learning

Zoubin Ghahramani

Gatsby Computational Neuroscience Unit

University College London

London WC1N 3AR, England

http://www.gatsby.ucl.ac.uk/~zoubin/

zoubin@gatsby.ucl.ac.uk

INTRODUCTION

"Graphical models" combine graph theory and probability theory to provide a general framework for representing models in which a number of variables interact. Graphical models trace their origins to many different fields and have been applied in wide variety of settings: for example, to develop probabilistic expert systems, to understand neural network models, to infer trait inheritance in geneologies, to model images, to correct errors in digital communication, or to solve complex decision problems. Remarkably, the same formalisms and algorithms can be applied to this wide range of problems.

Each node in the graph represent a random variable (or more generally a set of random variables). The pattern of edges in the graph represents the qualitative dependencies between the variables; the absence of an edge between two nodes means that any statistical dependency between these two variables is mediated via some other variable or set of variables. The quantitative dependencies between variables which are connected via edges are specified via parameterized conditional distributions, or more generally non-negative "potential functions". The pattern of edges and the potential functions together specify a joint probability distribution over all the variables in the graph. We refer to the pattern of edges as the *structure* of the graph, while the parameters of the potential functions simply as the *parameters* of the graph. In this chapter, we assume that the structure of the graph is given, and that our goal is to learn the parameters of the graph from data. Solutions to the problem of learning the graph structure from data are given in GRAPHICAL MODELS, STRUCTURE LEARNING.

We briefly review some of the notation from PROBABILISTIC INFERENCE IN GRAPHICAL MODELS which we will need to cover parameter learning in graphical models; we assume that the reader is familiar with the contents that chapter. More in-depth treatments of graphical models can be found in Pearl (1988), Heckerman (1996), Jordan (1998), and Cowell et al. (1999).

There are two main varieties of graphical model. *Directed graphical models*, also known as BAYESIAN NETWORKS, represent the joint distribution of d random variables $\mathbf{X} = (X_1, \dots, X_d)$ by a directed acyclic graph in which each node i, representing variable X_i, receives directed edges from its set of parent nodes π_i. The semantics of a directed graphical models are that the joint distribution of $\mathbf{X}$ can be factored into the product of conditional distributions of each variable given its parents. That is, for each setting $\mathbf{x}$ of the variable $\mathbf{X}$:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{d} p(x_i|\mathbf{x}_{\pi_i}, \boldsymbol{\theta}_i). \tag{1}$$

This factorization formalizes the graphical intuition that X_i depends on its parents: given its parents, X_i is statistically independent of all other variables which are not descendendents of X_i. The set of parameters governing the conditional distribution which relates $\mathbf{X}_{\pi_i}$ to X_i is denoted by $\boldsymbol{\theta}_i$, while the set of all parameters in the graphical model is denoted $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d)$. Note that 1 is identical to equation 1 in PROBABILISTIC INFERENCE IN GRAPHICAL MODELS except that we have made explicit the dependence of the conditional distributions on the model parameters.

Undirected graphical models represent the joint distribution of a set of variables via a graph with undirected edges. Defining $\mathcal{C}$ to be the set of maximal cliques (i.e. fully-connected subgraphs) of this graph, an undirected graphical model corresponds to the statement that the joint distribution of $\mathbf{X}$ can be factored into the product of functions over the variables in each clique:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C|\boldsymbol{\theta}_C) \tag{2}$$

where $\psi_C(\mathbf{x}_C|\boldsymbol{\theta}_C)$ is a potential function assigning a non-negative real number to each configuration $\mathbf{x}_C$ of $\mathbf{X}_C$, and is parameterized by $\boldsymbol{\theta}_C$. An undirected graphical model corresponds to the graphical intuition that dependencies are transmitted via the edges in the graph: each variable is statistically independent of all other variables given the set of variables it is connected to (i.e. its neighbors). Note again that we have reproduced equation 2 from PROBABILISTIC INFERENCE IN GRAPHICAL MODELS while making explicit the parameters of the potential functions.

The chapter is organized as follows. We start by concentrating on directed graphical models. In the next section, we discuss the problem of learning maximum likelihood (ML) parameters when all the variables are observed. The following secion generalizes this to the case where some of the variables are hidden or missing, and introduces the EM algorithm. We then turn to learning parameters of undirected graphical models using both EM and IPF. Finally, we discuss the Bayesian approach in which a posterior distribution over parameters is inferred from data.

ML LEARNING FROM COMPLETE DATA

Assume we are given a data set $\mathbf{d}$ of N independent and identically distributed observations of the settings of all the variables in our directed graphical model $\mathbf{d} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, where $\mathbf{x}^{(n)} = (x_1^{(n)}, \ldots, x_d^{(n)})$. The *likelihood* is a function of the parameters which is proportional to the probability of the observed data:

$$p(\mathbf{d}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}^{(n)}|\boldsymbol{\theta}) \tag{3}$$

We assume that the parameters are unknown and we wish to estimate them from data. We focus on the problem of estimating a single setting of the parameters which maximizes the likelihood (3). (In contast, the Bayesian approach to learning described in the last section starts with a prior distribution over the parameters $p(\boldsymbol{\theta})$ which is meant to capture any prior knowedge we may have about $\boldsymbol{\theta}$, and infers the posterior distribution over parameters given the data using Bayes rule.) Equivalently we can maximize the log likelihood:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \log p(\mathbf{d}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}|\boldsymbol{\theta}) && (4) \\ &= \sum_{n=1}^{N} \sum_{i=1}^{d} \log p(x_i^{(n)}|\mathbf{x}_{\pi_i}^{(n)}, \boldsymbol{\theta}_i) && (5) \end{aligned}$$

where the last equality makes use of the factorization (1) of joint distribution in the directed graphical model. If we assume that the parameters $\boldsymbol{\theta}_i$ governing the conditional probability distribution of X_i given its parents are distinct and functionally independent of the parameters governing the conditional probability distribution of other nodes in the graphical model, then the log likelihood decouples into a sum of local terms involving each node and its parents:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{d} \mathcal{L}_i(\boldsymbol{\theta}_i) \tag{6}$$

where $\mathcal{L}_i(\boldsymbol{\theta}_i) = \sum_n \log p(x_i^{(n)}|\mathbf{x}_{\pi_i}^{(n)}, \boldsymbol{\theta}_i)$. Each $\mathcal{L}_i$ can be maximized independently as a function of $\boldsymbol{\theta}_i$. For example, if the $\mathbf{X}$ variables are discrete and $\boldsymbol{\theta}_i$ is the conditional probability table for x_i given its parents, then the ML estimate of $\boldsymbol{\theta}_i$ is simply a normalised table containing counts of each setting of X_i given each setting of its parents in the data set.

Maximum a posteriori (MAP) parameter estimation incorporates prior knowledge about the parameters in the form of a distribution $p(\boldsymbol{\theta})$. The goal of MAP estimation is to find the parameter setting that maximizes the posterior over parameters, $p(\boldsymbol{\theta}|\mathbf{d})$, which is proportional to the prior times the likelihood. If the prior factorizes over the parameters governing each conditional probability distribution, i.e. $p(\boldsymbol{\theta}) = \prod_i p(\boldsymbol{\theta}_i)$ then MAP estimates can be found by maximizing

$$\mathcal{L}'(\boldsymbol{\theta}) = \sum_{i=1}^{d} \mathcal{L}_i(\boldsymbol{\theta}_i) + \log p(\boldsymbol{\theta}_i). \tag{7}$$

The log prior can be seen as a regularizer, which can help reduce overfitting in situations where there is insufficient data for the parameters to be well-determined (see GENERALIZATION AND REGULARIZATION IN NONLINEAR LEARNING SYSTEMS). While ML estimation is invariant to reparameterization, since the location of the maximum of the likelihood function does not change if you apply a one-to-one transformation $f : \boldsymbol{\theta} \rightarrow \phi$, MAP estimation is not. Indeed, for *any* $\tilde{\boldsymbol{\theta}}$ one can always find one-to-one mapping such that the MAP estimate of $\phi = f(\tilde{\boldsymbol{\theta}})$, as long as $p(\boldsymbol{\theta}|\mathbf{d}) > 0$ in a small neighborhood around $\tilde{\boldsymbol{\theta}}$. Thus, care should be taken in the choice of parameterization.

ML LEARNING WITH HIDDEN VARIABLES

The EM algorithm

Often, the observed data will not include the values of some of the variables in the graphical model. We refer to these variables as missing or hidden variables. With hidden variables, the log likelihood cannot be decomposed as in (6). Rather, we find:

$$\mathcal{L}(\boldsymbol{\theta}) = \log p(\mathbf{x}|\boldsymbol{\theta}) = \log \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) \tag{8}$$

where $\mathbf{x}$ denotes the setting of the observed variables, $\mathbf{y}$ the setting of the hidden variables, and $\sum_{\mathbf{y}}$ is the sum (or integral) over $\mathbf{Y}$ required to obtain the marginal probability of the observed data. (For notational convenience, we have dropped the superscript (n) in (8) by evaluating the log likelihood for a single observation.) Maximising (8) directly is often difficult because the log of the sum can potentially couple all of the parameters of the model. We can simplify the problem of maximising $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ by making use of the following insight. Any distribution $q(\mathbf{Y})$ over the hidden variables defines a *lower bound* on $\mathcal{L}$:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) = \log \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) &= \log \sum_{\mathbf{y}} q(\mathbf{y}) \frac{p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})}{q(\mathbf{y})} && (9) \\
&\geq \sum_{\mathbf{y}} q(\mathbf{y}) \log \frac{p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})}{q(\mathbf{y})} && (10) \\
&= \sum_{\mathbf{y}} q(\mathbf{y}) \log p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) - \sum_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y}) && (11) \\
&= \mathcal{F}(q, \boldsymbol{\theta}) && (12)
\end{aligned}$$

where the inequality is known as Jensen's inequality and follows from the fact that the log function is concave. If we define the *energy* of a global configuration $(\mathbf{x}, \mathbf{y})$ to be $\log p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})$, then some readers may notice that the lower bound $\mathcal{F}(q, \boldsymbol{\theta}) \leq \mathcal{L}(\boldsymbol{\theta})$ is the negative of a quantity known in statistical physics as the *free energy*: the expected energy under q minus the entropy of q (Neal and Hinton, 1998). The Expectation-Maximization (EM) algorithm (Baum et al., 1970; Dempster et al., 1977) alternates between maximising $\mathcal{F}$ with respect to q and $\boldsymbol{\theta}$, respectively, holding the other fixed. Starting from some initial parameters $\boldsymbol{\theta}_0$, the $k+1$st iteration of the algorithm consists of the following two steps:

$$\textbf{E step:} \qquad q_{[k+1]} \leftarrow \arg\max_{q} \; \mathcal{F}(q, \boldsymbol{\theta}_{[k]}) \tag{13}$$

$$\textbf{M step:} \qquad \boldsymbol{\theta}_{[k+1]} \leftarrow \arg\max_{\boldsymbol{\theta}} \; \mathcal{F}(q_{[k+1]}, \boldsymbol{\theta}) \tag{14}$$

It is easy to show that the maximum in the E step is obtained by setting $q_{[k+1]}(\mathbf{y}) = p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{[k]})$, at which point the bound becomes an equality: $\mathcal{F}(q_{[k+1]}, \boldsymbol{\theta}_{[k]}) = \mathcal{L}(\boldsymbol{\theta}_{[k]})$. This involves inferring the distribution over the hidden variables given the observed variables and the current settings of the parameters, $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{[k]})$. Algorithms which solve this inference problem are presented in PROBABILISTIC INFERENCE IN GRAPHICAL MODELS . These algorithms make use of the structure of the graphical model to compute the quantities of interest efficiently by passing local messages from each node to its neighbors. Exact inference results in the bound being satisfied, but is in general computationally intractable for multiply-connected graphical structures. Even these "efficient" message passing procedures can take exponential time to compute the exact solution in such cases. For such graphs, deterministic and Monte Carlo methods provide a tool for approximating the E step of EM. One deterministic approximation which can be used in intractable models is to increase but not fully maximize the functional w.r.t. q in the E step. In particular, if q is chosen to be in a tractable family of distributions $\mathcal{Q}$ (i.e. a family of distributions for which the required expectations can be computed in polynomial time) then maximizing $\mathcal{F}$ over this tractable family

$$\textbf{E step:} \qquad q_{[k+1]} \leftarrow \arg\max_{q \in \mathcal{Q}} \; \mathcal{F}(q, \boldsymbol{\theta}_{[k]}) \tag{15}$$

is called a *variational approximation* to the EM algorithm (Jordan et al., 1998). This maximizes a lower bound to the likelihood rather than the likelihood itself.

The maximum in the M step is obtained by maximizing the first term in (11); since the entropy of q does not depend on $\boldsymbol{\theta}$:

$$\textbf{M step:} \qquad \boldsymbol{\theta}_{[k+1]} \leftarrow \arg\max_{\boldsymbol{\theta}} \; \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{[k]}) \log p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}). \tag{16}$$

This is the expression most often associated with the EM algorithm (Dempster et al., 1977), but it obscures the elegant interpretation of EM as coordinate ascent in $\mathcal{F}$. Since $\mathcal{F} = \mathcal{L}$ at the beginning of each M step (following an exact E step), and since the E step does not change $\boldsymbol{\theta}$, we are guaranteed not to decrease the likelihood after each combined EM step.

It is worthwhile to point out that it is usually not necessary to explicitly evaluate the entire posterior distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{[k]})$. Since $\log p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})$ contains both hidden and observed variables in the network, it can be factored as before as the sum of log probabilities of each node given its parents (5). Consequently, the quantities required for the M step are the expected values, under the posterior distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{[k]})$, of the same quantities (namely the *sufficient statistics*) required for ML estimation in the complete data case.

Consider a directed graphical with discrete variables, some hidden and some observed. Each node is parameterized by a conditional probability table which relates its values to the values of its parents. For example, if node i has two parents, j and k, and each variable can take on L values, then $\boldsymbol{\theta}_i$ is an $L \times L \times L$ table, with entries $\theta_{i,rst} = P(X_i = r|X_j = s, X_k = t)$. In the complete data setting where X_i, X_j, X_k are observed, the maximum likelihood estimate is:

$$\hat{\theta}_{i,rst} = \frac{\#(X_i = r, X_j = s, X_k = t)}{\#(X_j = s, X_k = t)} \tag{17}$$

where $\#(\cdot)$ denotes the count (frequency) with which the bracketed expression occurs in the data. However, if all three variables were hidden, then one could use the EM algorithm for learning the directed graphical model (Lauritzen, 1995; Russell et al., 1995). The analogous M step for $\theta_{i,rst}$ would be:

$$\hat{\theta}_{i,rst} = \frac{\sum_n P(Y_i = r, Y_j = s, Y_k = t|\, \mathbf{X} = \mathbf{x}^{(n)})}{\sum_n P(Y_j = s, Y_k = t|\, \mathbf{X} = \mathbf{x}^{(n)})}. \tag{18}$$

The sufficient statistics of the data required to estimate the parameters are the counts of the setting of each node and its parents—no other information in the data is relevant for ML parameter estimation. The *expectation* step of EM computes the expected value of these sufficient statistics.

The EM algorithm provides an intuitive way of dealing with hidden/missing data. The E step "fills-in" the hidden variables with the distribution given by the current model. The M step then treats these filled-in values as if they had been observed and re-estimates the model parameters. It is pleasantly surprising that these steps result in a convergent procedure for finding the most likely parameters. Although EM is intuitive and often easy to implement, it is sometimes not the most efficient algorithm for finding ML parameters.

The EM procedure for learning directed graphical models with hidden variables can be applied to a wide variety of well-known models. Of particular note is the special case known as the Baum-Welch algorithm for training HIDDEN MARKOV MODELS (Baum et al., 1970; Rabiner, 1989). In the E step it uses a local message passing algorithm called the forward-backward algorithm to compute the required expected sufficient statistics. In the M step it uses parameter restimation equation based on expected counts analogous to (18). The Baum-Welch algorithm predates the original EM algorithm and in fact the convergence proof for EM is based on the one given in (Baum et al., 1970). The EM algorithm can also be used to fit a variety of other models that have been studied in the machine learning, neural networks, statitistics and engineering literatures. These include linear dynamical systems, factor analysis, mixtures of Gaussians, and mixtures of experts (see Roweis and Ghahramani 1999 for a review). It is straightforward to modify EM so that it maximizes the parameter posterior probability rather than the likelihood.

PARAMETER LEARNING IN UNDIRECTED GRAPHICAL MODELS

When compared to learning the parameters of directed graphical models, undirected graphical models present an additional challenge: the partition function. Even if each clique has distinct and functionally independent parameters, the partition function from (2)

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C | \boldsymbol{\theta}_C) \tag{19}$$

couples all the parameters together. We examine the effect this coupling has in the context of an undirected graphical model which has has a great deal of impact in the neural networks field: the Boltzmann machine (Ackley et al., 1985).

Boltzmann machines (BMs) are indirected graphical models over a set of d binary variables $S_i \in \{0, 1\}$ (see also SIMULATED ANNEALING AND BOLTZMANN MACHINES). The probability distribution over the variables in a BM is given by

$$P(\mathbf{s}|W) = \frac{1}{Z(W)} \exp\left\{\frac{1}{2}\sum_{i=1}^{d} \sum_{j \in \mathrm{ne}(i)} W_{ij} s_i s_j\right\} = \frac{1}{Z(W)} \prod_{(ij)} \exp\left\{W_{ij} s_i s_j\right\}. \tag{20}$$

The first equation uses standard notation for BMs, where W is the symmetric matrix of weights (i.e. model parameters) and ne(i) is the set of neighbors of node i in the BM. The second equation writes it as a product of clique potentials where (ij) denotes the clique consisting of the pair of connected nodes i and j. (Actually, in BMs, the maximal cliques in the graph may be very large although the interactions are all pairwise; because of these pairwise constraint on interactions we abuse terminology and consider the "cliques" to be the pairs, no matter what the graph connectivity is.)

Assuming that S_i and S_j are observed, taking derivatives of the log probability of the n^{th} data point with respect to W_{ij},

$$\frac{\partial \log P(\mathbf{s}^{(n)}|W)}{\partial W_{ij}} = s_i^{(n)} s_j^{(n)} - \sum_s s_i s_j P(\mathbf{s}|W) = \langle s_i s_j \rangle_n^+ - \langle s_i s_j \rangle^-, \tag{21}$$

we find that it is the of difference between the correlation of S_i and S_j in the data and the correlation of S_i and S_j in the model (the $\langle \cdot \rangle$ notation means expectation). The standard ML gradient descent learning rule for Boltzmann machines therefore tries to make the model match the correlations in the data. The same learning rule applies if there are hidden variables.

The second term arises from the partition function. Note that even for fully observed data, while the first term can be computed directly from the observed data, the second term depends potentially on all the parameters, underlining the fact that the partition function couples the parameters in undirected models. Even for fully observed data, computing the second term is nontrivial; for fully connected BMs it is intractable and needs to be approximated.

The IPF algorithm

Consider the following problem: given an undirected graphical model, and an initial set of clique potentials, we wish to find the clique potentials closest to the initial potentials that satisfy a certain set of consistent marginals. Closeness of probability distributions in this context is measured using the Kullback-Leibler divergence (Cover and Thomas, 1991). The simplest example is a clique of two discrete variables, X_i and X_j , where the clique potential is proportional to the continency table for the joint probability of these variables $P(X_i, X_j)$, and the marginal constraints are $P(x_i) = \hat{P}(x_i)$ and $P(x_j) = \hat{P}(x_j)$. These constraints could, for example, have come from observing data with these marginals. A very simple and intuitive iterative algorithm for trying to satisfy these constraints is to start from the initial table and satisfy each constraint in turn. For exampling if we want to satisfy the marginal on X_i:

$$P_k(x_i, x_j) \quad = \quad P_{k-1}(x_i, x_j)\frac{\hat{P}(x_i)}{P_{k-1}(x_i)} \tag{22}$$

This has to be iterated over X_i and X_j since satisfying one marginal can change the other marginal. The simple algorithm is known as Iterative Proportional Fitting (IPF), and it can be generalized in several ways (Darroch and Ratcliff, 1972).

More generally we wish to find a distribution in the form

$$p(\mathbf{x}) = \frac{1}{Z}\prod_c \psi_c(\mathbf{x}_c) \tag{23}$$

which minimizes the KL divergence to some prior $p_0(\mathbf{x})$ and satisfies a set of constraints (the data) of the form:

$$\sum_{\mathbf{x}} a_r(\mathbf{x})p(\mathbf{x}) = h_r, \tag{24}$$

where r indexes the constraint. If the prior is set to the uniform distribution, and the constraints are measured marginal distributions over all the variables in each of the cliques of the graph, then the problem solved by IPF is equivalent to finding the maximum likelihood clique potentials given a complete data set of observations.

IPF can be used to train an ML Boltzmann machine if $\hat{P}(S_i, S_j)$ is given by the data set for all pairs of variables connected in the BM. The procedure is to start from the uniform distribution (i.e. all weights set to 0) then apply IPF steps to each clique potential until all marginals match those in the data. One can generalize this by starting from a non-uniform distribution, which would give a BM with minimum divergence from the starting distribution.

But what if some of the variables in the BM are hidden? Byrne (1992) presents an elegant solution to this problem using ideas from Alternating Minimization (AM) and information geometry (Csizár and Tusnády, 1984). One step of the alternating minimization computes the distribution over the hidden variables of the BM given the observed variables. This is the E step of EM, and can also be interpreted within information geometry as finding the probability distribution which satisfies the marginal constraints and is closest to the space of probability distributions defined by the Boltzmann machines. The other step of the minimization starts from $W = 0$ (the maximum entropy distribution for a BM), and uses IPF to find the BM weights which satisfy all the marginals found in the E step, $\hat{P}_{ij} = \langle s_i s_j \rangle^+$. This is the M step of the algorithm; a single M step thus involves an entire IPF optimization. The update rule for each step of the IPF optimization for a particular weight is:

$$W_{ij} \leftarrow W_{ij} + \log \left[\frac{\hat{P}_{ij}}{\langle s_i s_j \rangle^-} \frac{(1 - \langle s_i s_j \rangle^-)}{(1 - \hat{P}_{ij})} \right] \tag{25}$$

This algorithm is conceptually interesting as it presents an alternative method for fitting BMs with ties to IPF and AM procedures. However, it is impractical for large multiply-connected Boltzmann machines since computing the exact unclamped correlations $\langle s_i s_j \rangle^-$ can take exponential time.

Although we have presented this EM-IPF algorithm for the case of Boltzmann machines, it is widely applicable to learning many undirected graphical models. In particular, in the E step, marginal distributions over the set of variables in each clique in the graph are computed conditioned on the settings of the observed variables. A propagation algorithm such as the Junction Tree algorithm can be used for this step (PROBABILISTIC INFERENCE IN GRAPHICAL MODELS). In the M step the IPF procedure is run so as to satisfy all the marginals computed in the E step. There also exist Junction Tree style propagation algorithms which exploit the structure of the graphical model to solve the IPF problem efficiently (Jiroušek and Přeučil, 1995; Teh and Welling, 2002).

BAYESIAN LEARNING OF PARAMETERS

A Bayesian approach to learning starts with some *a priori* knowledge about the model structure—the set of arcs in the Bayesian network—and model parameters. This initial knowledge is represented in the form of a prior probability distribution over model structures and parameters, and is updated using the data to obtain a posterior probability distribution over models and parameters. In this article we will assume that the model structure is given and we focus on computing the posterior probability distribution over parameters (see GRAPHICAL MODELS, STRUCTURE LEARNING for solutions to the problem of inferring model structure, and also BAYESIAN METHODS FOR NEURAL NETWORKS).

For a given model structure, $\mathbf{m}$, we can compute the posterior distribution over the parameters:

$$p(\boldsymbol{\theta}|\mathbf{m}, \mathbf{d}) = \frac{p(\mathbf{d}|\boldsymbol{\theta}, \mathbf{m}) p(\boldsymbol{\theta}|\mathbf{m})}{p(\mathbf{d}|\mathbf{m})}. \tag{26}$$

If the data set is $\mathbf{d} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ and we wish to predict the next observation, $\mathbf{x}^{(n+1)}$ based on our data and model, then the Bayesian prediction

$$p(\mathbf{x}^{(n+1)}|\mathbf{d}, \mathbf{m}) = \int p(\mathbf{x}^{(n+1)}|\boldsymbol{\theta}, \mathbf{m}, \mathbf{d})\ p(\boldsymbol{\theta}|\mathbf{m}, \mathbf{d})\ d\boldsymbol{\theta} \tag{27}$$

averages over the uncertainty in the model parameters. This is known as the *predictive distribution* for the model.

In the limit of a large data set and as long as the prior over the parameters assigns non-zero probability in the region around the maximum likelihood parameter values, the posterior $p(\boldsymbol{\theta}|\mathbf{m}, \mathbf{d})$ will be sharply peaked around the maxima of the likelihood, and therefore the predictions of a single maximum likelihood (ML) model will be similar to those obtained by Bayesian integration over the parameters.

Often models are fit with relatively small amounts of data, so asymptotic results are not applicable and the predictions of the ML estimate will differ significantly from those of Bayesian averaging. In such sitations it is important to compute or approximate the averaging over parameters in (27). For certain discrete models with Dirichlet distributed priors over the parameters, and for certain linear-Gaussian models, it is possible to compute these integrals exactly. Otherwise, approximations can be used. There are a large number of approximations to the integral over the parameter distribution that have been used in graphical models, include Laplace's approximation, variational approximations (Attias, 1999; Ghahramani and Beal, 2000), the expectation-propagation algorithm (Minka, 2001) and a variety of MCMC methods (see Neal 1993 for a review).

DISCUSSION

There are several key insights regarding parameter learning in graphical models. When there are no hidden variables in a directed graphical model, the graph structure determines the statistics of the data needed to learn the parameters: the joint distribution of each variable and its parents in the graph. Parameter estimation can then often occur independently for each node. The presence of hidden variables introduces dependencies between the parameters. However, the EM algorithm transforms the problem so that in each M step the parameters of the graphical model are again uncoupled. The E step of EM "fills-in" the hidden variables with the distribution predicted by the model, thereby turning the hidden-data problem into a complete-data problem.

The intuitive appeal of EM has lead to its widespread use in models of unsupervised learning where the goal is to learn a generative model of sensory data. The E step corresponds to perception or recognition: inferring the (hidden) state of the world from the sensory data; while the M step corresponds to learning: modifying the model that relates the actual world to the sensory data. It has been suggested that top-down and bottom-up connections in cortex play the roles of generative and recognition models (see HELMHOLTZ MACHINES AND WAKE SLEEP LEARNING). From a graphical model perspective, the bottom-up recognition model in Helmholtz machines can be thought of as a graph which approximates the distribution of the hidden variables given the observed variables, much in the same way as the variational approximation approximates that same distribution.

Undirected graphical models pose additional challenges. The partition function is usually a function of the parameters, and can introduce dependencies between the parameters even in the complete data case. The IPF algorithm can be used to fit undirected graphical models from complete data, and an IPF-EM algorithm can be used when there is hidden data as well.

Although maximum likelihood learning is adequate when there is enough data, in general it is necessary to approximate the average over the parameters. Averaging avoids overfitting; it is hard to see how overfitting can occur when nothing is "fit" to the data. Non-Bayesian methods for avoiding overfitting often also involve averaging, for example, via boostrap resampling of the data. Even with parameter averaging, predictions can suffer in quality if the assumed structure of the model—the conditional indepenence relationships—are incorrect. To overcome this, it is necessary to generalize the approach in this chapter to learn the structure of the model as well as the parameters from data. This topic is covered in GRAPHICAL MODELS: STRUCTURE LEARNING.

REFERENCES

Ackley, D., Hinton, G., and Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169.

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational Bayes. In *Proc. 15th Conf. on Uncertainty in Artificial Intelligence*.

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41:164–171.

Byrne, W. (1992). Alternating minimization and Boltzmann machine learning. *IEEE Trans on Neural Networks*, 3(4):612–620.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley, New York.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York, NY.

Csizár, I. and Tusnády, G. (1984). Information geometry and alternating minimization procedures. In Dudewicz, E. J., Plachky, D., and Sen, P. K., editors, *Statistics and Decisions, Suplementary Issue Number 1*, pages 205–237. Oldenbourg Verlag, Munich.

Darroch, J. N. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B*, 39:1–38.

Ghahramani, Z. and Beal, M. J. (2000). Propagation algorithms for variational Bayesian learning. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13, pages 507–513, Cambridge, MA. MIT Press.

Heckerman, D. (1996). A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06 [ftp://ftp.research.microsoft.com/pub/tr/TR-95-06.PS] , Microsoft Research.

Jiroušek, R. and Přeučil, S. (1995). On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics and Data Analysis*, 19:177–189.

Jordan, M. I., editor (1998). *Learning in Graphical Models*. Kluwer Academic Publishers.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An Introduction to variational methods in graphical models. In Jordan, M. I., editor, *Learning in Graphical Models*. Kluwer Academic Publishers.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 362–369, San Francisco, CA. Morgan Kaufmann Publishers.

Neal, R. M. (1993). Probabilistic inference using Markov chain monte carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Neal, R. M. and Hinton, G. E. (1998). A new view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*. Kluwer Academic Press.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–86.

Roweis, S. T. and Ghahramani, Z. (1999). A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345.

Russell, S. J., Binder, J., Koller, D., and Kanazawa, K. (1995). Local learning in probabilistic models with hidden variables. In *Proc. Intl. Joint Conf. on Artificial Intelligence*, pages 1146–1152.

Teh, Y. W. and Welling, M. (2002). The unified propagation and scaling algorithm. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*.