Graphical Models

Foundations of Neural Computation

edited by

Michael I. Jordan

Terrence J. Sejnowski

Graphical Models: Foundations of Neural Computation

Edited by Michael I. Jordan and Terrence J. Sejnowski

A Bradford Book

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Palatino and printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Graphical models : foundations of neural computation / edited by Michael I. Jordan and Terrence J. Sejnowski.
p. cm. — (A Bradford book) (Computational neuroscience)
ISBN 0-262-60042-0 (pbk. : alk. paper)
1. Neural networks (Computer science) 2. Computer graphics. I. Jordan, Michael Irwin, 1956– II. Sejnowski, Terrence J. (Terrence Joseph) III. Series. IV. Bradford book
QA76.87 .G72 2001
006.3′2—dc21

2001030212

Series Foreword

Computational neuroscience is an approach to understanding the information content of neural signals by modeling the nervous system at many different structural scales, including the biophysical, the circuit, and the systems levels. Computer simulations of neurons and neural networks are complementary to traditional techniques in neuroscience. This book series welcomes contributions that link theoretical studies with experimental approaches to understanding information processing in the nervous system. Areas and topics of particular interest include biophysical mechanisms for computation in neurons, computer simulations of neural circuits, models of learning, representation of sensory information in neural networks, systems models of sensory-motor integration, and computational analysis of problems in biological sensing, motor control, and perception.

Terrence J. Sejnowski
Tomaso A. Poggio

Sources

Smyth, P., Heckerman, D., and Jordan, M. I. 1997. Probabilistic independence networks for hidden Markov probability models. *Neural Computation* **9**(2), 227–269.

Hinton, G. E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations,* D. E. Rumelhart and J. L. McClelland, eds., pp. 282–317. MIT Press, Cambridge.

Saul, L., and Jordan, M. I. 1994. Learning in Boltzmann trees. *Neural Computation* **6**(6), 1174–1184.

Hinton, G. E. 1989. Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation* **1**(1), 142–150.

Saul, L. K., and Jordan, M. I. 2000. Attractor dynamics in feedforward neural networks. *Neural Computation* **12**(6), 1313–1335.

Kappen, H. J., and F. B. Rodríguez. 1998. Efficient learning in Boltzmann machines using linear response theory. *Neural Computation* **10**(5), 1137–1156.

Neal, R. 1992. Asymmetric parallel Boltzmann machines are belief networks. *Neural Computation* **4**(6), 832–834.

Frey, B. J., and Hinton, G. E. 1999. Variational learning in nonlinear Gaussian belief networks. *Neural Computation* **11**(1), 193–213.

Tipping, M. E., and Bishop, C. M. 1999. Mixtures of probabilistic principal component analyzers. *Neural Computation* **11**(2), 443–482.

Attias, H. 1999. Independent factor analysis. *Neural Computation* **11**(4), 803–851.

Jordan, M. I., and Jacobs, R. A. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* **6**(2), 181–214.

Krogh, A., and Riis, S. K. 1999. Hidden neural networks. *Neural Computation* **11**(2), 541–563.

Ghahramani, Z., and Hinton, G. E. 2000. Variational learning for switching state-space models. *Neural Computation* **12**(4), 831–864.

Tresp, V., and Hofmann, R. 1998. Nonlinear time-series prediction with missing and noisy data. *Neural Computation* **10**(3), 731–747.

Weiss, Y. 2000. Correctness of local probability propagation in graphical models with loops. *Neural Computation* **12**(1), 1–41.

Introduction

A "graphical model" is a type of probabilistic network that has roots in several different research communities, including artificial intelligence (Pearl 1988), statistics (Lauritzen 1996), and neural networks (Hertz, Krogh, and Palmer 1991). The graphical models framework provides a clean mathematical formalism that has made it possible to understand the relationships among a wide variety of network-based approaches to computation and, in particular, to understand many neural network algorithms and architectures as instances of a broader probabilistic methodology. Moreover, this formal framework has made it possible to identify those features of neural network algorithms and architectures that are novel and to extend them to other more general graphical models. This interplay between the general formal framework of graphical models and the exploration of new algorithms and architectures is exemplified in the chapters included in this volume. These chapters, chosen from *Neural Computation,* include many foundational papers of historical importance as well as papers that are at the research frontier. The volume is intended for a broad range of students, researchers, and practitioners who are interested in understanding the basic principles underlying graphical models and in applying them to practical problems.

Probabilistic and information-theoretic approaches have become dominant in the neural network literature, as researchers have attempted to formalize "adaptivity" in neural computation and understand why some adaptive methods perform better than others. The probabilistic framework has also provided a guide in the exploration of new algorithms and architectures. At the same time, the notion of "locality" has continued to exert a key constraint on neural network research, restricting the kinds of architectures and algorithms that are studied. The particular relevance of graphical models to this research effort is that the graphical models framework provides formal definitions of both adaptivity and locality. It does so by forging a mathematical link between probability theory and graph theory.

Graphical models use graphs to represent and manipulate joint probability distributions. The graph underlying a graphical model may be directed, in which case the model is often referred to as a *belief network* or a *Bayesian network,* or the graph may be undirected, in which case the model is generally referred to as a *Markov random field.* A graphical model has both a structural component—encoded by the pattern of edges in the graph—and a parametric component—encoded by numerical "potentials" associated with sets of edges in the graph. The relationship between these components underlies the computational machinery associated with graphical models. In particular, general *inference algorithms* allow statistical quantities (such as likelihoods and conditional probabilities) and information-theoretic quantities (such as mutual information and conditional entropies) to be computed efficiently. *Learning algorithms* build on these inference algorithms and allow parameters and structures to be estimated from data. All of these probabilistic computations make use of data structures associated with the graph—in particular, an important data structure known as a *junction tree* (see chapter 1). The junction tree groups nodes into clusters and defines probabilistic "messages" that pass among the clusters; this essentially amounts to a graph-theoretic characterization of computational "locality" for general probabilistic inference.

Many neural network architectures, including essentially all of the models developed under the rubric of "unsupervised learning" (Hinton and Sejnowski 1999) as well as supervised Boltzmann machines, mixtures of experts, and normalized radial basis function networks, are special cases of the graphical model formalism, both architecturally and algorithmically. Many other neural networks, including the classical multilayer perceptron, can be profitably analyzed from the point of view of graphical models.

The graphical model literature in AI and statistics has contributed the general formalism for understanding relationships between graphs and probabilities; the neural network literature has contributed a rather far-flung exploration in the space of architectures and algorithms. This exploration is the principal subject matter of this book. Before turning to specific examples, let us give a brief overview of some of the general themes that have characterized this exploration.

Classical graphical model architectures in AI have often used localist representations, in which a single node represents a complex concept, such as "chair." Neural network researchers, on the other hand, have explored *distributed* or *factorial* representations in which single nodes represent simpler properties that are broadly tuned and overlapping. This has allowed much larger problems to be tackled. Also, the graphical model literature has generally focused on exact probabilistic inference or sampling-based inference methods, whereas neural network research has studied a wider class of approximate inference methods. For example, the *mean-field*, or *variational*, methodology developed first for the Boltzmann machine and related undirected models, has flowed into the general graphical formalism and yielded fast new algorithms for approximate probabilistic inference. Finally, the neural network literature has delved deeply into nonlinear classification and regression, contributing a variety of new methods for parameter estimation and regularization in graphical models, with a particular focus on "on-line algorithms" (which can be viewed as defining another, temporal, notion of "locality"). In general, the bi-directional flow of ideas between these two fields has led to a significantly broader understanding of network-based computation. This understanding is reflected and pursued in the following chapters.

Readers new to the topic of graphical models should start with the first three sections of the chapter by Smyth, Heckerman, and Jordan (chapter 1), which provide a short overview. A full presentation can be found in any of several recent textbooks (e.g., Cowell et al. 1999). See also Jordan (1999) for several tutorial articles that provide basic background for the chapters presented here.

The Boltzmann Machine

The *Boltzmann machine* (chapter 2) is a probabilistic network of binary nodes. Historically, the Boltzmann machine played an important role in the development of the neural network field, as the first general multilayer architecture to employ hidden units between the input and output nodes. As we discuss in this section, the Boltzmann machine is a special case of an undirected graphical model, or *Markov random field (MRF).* The inference and learning algorithms for Boltzmann machines, and in particular the treatment of hidden units, exemplify more general solutions to the problem of inference and learning in graphical models with latent variables.

While general MRF's represent joint probability distributions as products of arbitrary local functions ("potentials") on the cliques of the graph,[1] the Boltzmann machine adopts a restricted parameterization in which the potentials are formed from *pairwise* factors. These pairwise factors take the form $\exp\{J_{ij}S_iS_j\}$, where J_{ij} is the weight on the edge between unit i and j, and S_i and S_j are the (binary) values of units i and j, respectively. (In a general MRF, higher-order interactions such as $J_{ijk}S_iS_jS_k$ would be included—when the nodes S_i, S_j, and S_k are in a clique, namely, are mutually interconnected). Taking products of these local potentials yields the total potential $\exp\{\sum_{i<j} J_{ij}S_iS_j\}$, which, when normalized, defines a Boltzmann distribution:

$$P(S) = \frac{e^{-E(S)}}{Z} \tag{1}$$

for a quadratic *energy function* $E(S) \overset{def}{=} -\sum_{i<j} J_{ij}S_iS_j$. From this joint probability distribution, we can define arbitrary conditional probabilities of one set of nodes given another set of nodes. Calculating these conditionals defines the *inference problem* for Boltzmann machines.

For general Boltzmann machines, in particular for the fully connected Boltzmann machines that have generally been studied in the literature, there are no structural properties (conditional independencies) to take advantage of, and the inference problem is intractable. Approximate inference techniques have generally been employed—in particular, stochastic sampling (Gibbs sampling) enhanced with simulated annealing. Although these methods do provide a way to study the Boltzmann

[1]A *clique* is a fully connected subgraph. A clique consisting of n binary nodes can be in one of 2^n configurations, where a *configuration* is an assignment of a binary value to each node in the clique. A *potential* is a function that assigns a nonnegative real number to each configuration.

machine empirically, they are slow and are generally viewed as complex (particularly when used in the setting of learning algorithms, where multiple simulated annealing passes are required). Historically, when the multilayer perceptron became popular, Boltzmann machines lost their luster.

The fact that the worst-case, fully connected Boltzmann machine presents no opportunities for fast inference does not imply that Boltzmann machines in general present no such opportunities. This point of view was emphasized by Saul and Jordan (chapter 3), who studied Boltzmann machines in which the hidden units form a tree. They showed that in such architectures it is not necessary to resort to Gibbs sampling to solve the inference problem; rather, a simple deterministic recursion known as *decimation* can be employed to calculate the conditional probabilities. The time required for the computation is proportional to the width of the graph. These are Boltzmann machines that can be "solved."

The decimation rule can be generalized beyond the pairwise interactions that characterize the classical Boltzmann machine, yielding an exact calculation method for general MRFs. Interestingly, this rule is a special case of the junction tree methodology that has been developed for inference in arbitrary graphical models (Cowell et al. 1999). There appears to be no particular advantage to the decimation approach, and indeed the junction tree approach has the advantage of providing an explicit method for estimating the time complexity of inference—the time complexity is exponential in the size of the largest clique in the *triangulated graph* of the network. Thus it is possible to identify systematically the classes of Boltzmann machines for which exact inference is efficient.

Mean Field Approximation

In 1987, Peterson and Anderson (1987) presented an alternative approach to inference for the Boltzmann machine that has had substantial impact. Their approach was based on an approximation known in physics as the "mean field" approximation. Under this approximation, the (approximate) mean value of the conditional probability distribution at each node is written as a function of the (approximate) mean values of its neighbors, and a so-called self-consistent set of mean values is obtained by iteratively evaluating these functions. For the Boltzmann machine, these iterative equations turn out to take a simple classical form in which each node's value is the logistic function of a weighted sum of its neighbors' values. These are the standard nonlinear equations proposed by Hopfield (1984) for the "continuous Hopfield network," and they can be shown—via Lyapunov theory—to be locally convergent (Cohen and Grossberg 1983; Hopfield 1984).

Peterson and Anderson's idea has been taken in two somewhat different directions. One line of research has focused on optimization problems, where the mean field approach has given rise to a general methodology known as *deterministic annealing* (Yuille and Kosowsky 1994). In deterministic annealing, the focus is on the energy function rather than the distribution that it defines; in particular, the goal is to find the minima of the energy function. (The probabilistic framework serves the subsidiary role of smoothing the energy function.) The mean field equations are generally derived from the point of view of saddle point approximation, which gives rise to a free "temperature" parameter that controls the degree of smoothing. In a procedure reminiscent of interior point methods, the mean field equations are solved for a gradually decreasing set of temperatures. It is possible to relate the limiting solution of these equations (as the temperature goes to zero) to the minima of the energy function (Elfadel 1995).

In a second branch of research, the focus has been on mean field theory as a methodology for approximation probabilistic inference in general graphical models. Here the emphasis has been on extending the basic approach to a wider class of architectures and on developing more refined versions of the approximation. A different point of view has proved to be fruitful in which the mean field approximation is viewed as the expression of a *variational principle*. Given a distribution $P(S)$ that is costly to calculate, approximate $P(S)$ by choosing a distribution $Q(S|\mu)$ from a family of approximating distributions, where the *variational parameter* μ indexes the family. The variational parameter is chosen so as to minimize the Kullback-Leibler (KL) divergence between Q and P:

$$\mu^* = \underset{\mu}{\operatorname{argmin}} \left\{ \sum_{\{S\}} Q(S|\mu) \ln \frac{Q(S|\mu)}{P(S)} \right\},$$

where the sum is taken over all configurations of S (assumed discrete for simplicity).

When $Q(S|\mu)$ is taken to be the completely factorized distribution, namely, $Q(S) = \prod_i Q(S_i|\mu_i)$, and when $P(S)$ is the Boltzmann distribution in equation (1), then one obtains the mean field equations of Peterson and Anderson. That is, the Peterson and Anderson equations arise by taking the derivative of the KL divergence with respect to μ_i and setting to zero.

Saul and Jordan (1996) observed that a wider class of approximations could be obtained by choosing a wider class of approximating distributions $Q(S|\mu)$. Note that a completely factorized $Q(S|\mu)$ corresponds to a subgraph of the original graphical model in which all edges are omitted. By considering subgraphs that retain some of the edges of the original graph, while maintaining tractability by restricting the subgraph to be a sparse graph (such as a chain or a tree), more refined variational approximations can be obtained. Moreover, in minimizing the KL divergence for such approximations, it is necessary to solve the inference problem for the tractable subgraph. Thus exact inference algorithms (such as the junction tree algorithm) become subroutines within an overall variational approximation. Several of the chapters included in the collection reflect this point of view.

There are other refinements to mean field theory that have been studied in the context of graphical models. Kappen and Rodríguez (chapter 6) studied the *linear response correction* (Parisi 1988) to the naive mean field approximation, which provides an improved approximation to the second-order statistics. Applying this correction to mean field equations for the Boltzmann machine, they found significant improvements in inferential accuracy.

Thus far we have focused on inference, but an equally important problem is that of learning the parameters of the model. In the setting of graphical models, the learning problem and the inference problem are closely related and learning algorithms generally make use of inference algorithms as an "inner loop." In the context of the Boltzmann machine, the classical approach is to use Gibbs sampling as an inner loop to obtain the statistics that are needed for the gradient descent procedure (the "outer loop"). There is, however, no reason to focus exclusively on sampling methods. For tractable architectures, it is preferable to calculate the necessary statistics exactly (using the junction tree algorithm). Alternatively, as shown by Hinton (chapter 4), the approximation provided by the mean field approach provides an appropriate inner loop for a gradient descent algorithm for learning. This idea has been taken further by Neal and Hinton (1999), who develop a link between approximate inference and approximate "E steps" for the EM algorithm. In general, approximate inference algorithms can be used to increase a (tractable) lower bound on an (intractable) likelihood.

Directed Graphical Models

Another point of contact between the neural network literature and the graphical model literature was made by Neal (chapter 7; see also Neal 1992 for a fuller presentation). Neal observed that certain so-called asymmetric Boltzmann machines are actually special cases of *directed graphical models*, also known as *belief networks* or *Bayesian networks*. Directed graphical models define their joint probabilities by taking products of local *conditional* probabilities. In many ways this yields a simpler entry point into the graphical model framework than the undirected formalism of Boltzmann machines.

The general definition of a joint probability distribution for a directed graphical model is given as follows:

$$P(S) = \prod_i P(S_i|\pi_i), \tag{2}$$

where π_i represents the set of parents of node S_i. Note in particular that there is no need for a normalizing constant Z in this approach to defining the joint probability.

For the special case of binary S_i, one interesting possibility is to take $P(S_i|\pi_i)$ to be the logistic function of a linear weighted sum of the parent nodes. This yields a directed graphical model known as a *sigmoid belief* *network*. As observed by Neal, a sigmoid belief network is a close cousin of the multilayer perceptron. Neal proposed using Gibbs sampling as the inferential engine for sigmoid belief networks, but, as in the case of the Boltzmann machine, for certain architectures (e.g., trees) one can perform exact inference efficiently using the junction tree algorithm.

Saul, Jaakkola, and Jordan (1996) derived the analog of the Peterson and Anderson mean field theory for sigmoid belief networks. The directed nature of the graph yields additional terms in the mean field equations that are not present in the undirected Boltzmann machine. Saul and Jordan (chapter 5) took this approach further in the context of large layered networks, where a central limit theorem expansion is justified. An interesting feature of their work is that the (approximate) maximum likelihood learning algorithm that they derive includes "weight-decay" terms that are familiar from statistically motivated regularization methods. Thus approximate inference based on a simplifying variational distribution can be preferable to exact inference for the purposes of parameter estimation.

Although most of the research on variational approximation algorithms has been carried out for networks of discrete nodes, Frey and Hinton (chapter 8) have developed variational algorithms for several kinds of continuous nodes. The derivation presented in their chapter includes piecewise linear nodes and nodes with continuous sigmoidal nonlinearities.

Latent Variable Models

The Boltzmann machine, sigmoid belief networks, and mean field or variational algorithms have provided a set of links to the graphical models literature; another link has been provided by mixture models and more general latent variable models. In the simplest cases these models are handled via exact inference methods, but in more complex cases the models shade into the layered graphical models discussed in the previous section, where sampling or variational methods are generally required.

Mixture models provide a probabilistic setting for the development of clustering algorithms, both unsupervised (Duda and Hart 1973; Nowlan 1990) and supervised (Jacobs et al. 1991). Each data point is assumed to be drawn from one of a fixed set of classes, but the class label is assumed to be "missing" or "latent" and must be inferred from the model.

As a graphical model, a classical mixture model for unsupervised clustering has a particularly simple representation (see figure 1). The unshaded ("hidden") node represents the class label, ω, and the shaded node represents the observed data point $\mathbf{y}$. The inference problem for this graphical model is that of calculating the probability of the hidden node given the observed node, namely, $P(\omega|\mathbf{y})$. The calculation of this posterior probability is the "inner loop" in a procedure (the expectation-

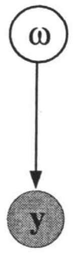

Figure 1: The graphical model representation for a mixture model, where the node labeled **y** represents an observed data point and the node labeled ω represents the (latent) class. The joint probability is given by $P(\omega)P(\mathbf{y}|\omega)$; marginalizing over ω yields a mixture.

maximization or "EM" algorithm) that estimates the parameters of the model.

An alternative latent variable model is provided by factor analysis (FA), where the underlying variable is a continuous rather than a discrete vector. This model can also be represented as the two-node graphical structure in figure 1 (although this representation hides the independencies among the components of the vectors). The model is parameterized by letting the latent variable be Gaussian with diagonal covariance matrix, and by letting the observed variable be Gaussian with a mean that is a linear function of the latent variable.

Factor analysis provides a technique for dimensionality reduction in which the data are assumed to lie near a low-dimensional hyperplane. A closely related model involving a probabilistic variant of principal component analysis (PCA) has been developed by Roweis (1998) and Tipping and Bishop (chapter 9); this model can also be represented as the two-node graphical structure in figure 1.

A number of authors, including Ghahramani and Hinton (1998), Hinton, Dayan, and Revow (1997), and Tipping and Bishop (chapter 9), have studied mixtures of FA or PCA models. We have reprinted the latter paper, which is representative of this line of research. The basic model can be rendered as a graphical model as shown in figure 2. As in a mixture model, the latent node ω is a discrete node representing the hidden class label. For each value of ω, we obtain a FA or PCA model, where the latent node y is a continuous node representing the FA or PCA subspace. Under this model, data are assumed to form clusters, where each cluster is represented as a lower-dimensional linear manifold.

Independent components analysis (ICA) is another latent variable model with links to FA and PCA (Comon 1994; Bell and Sejnowski 1995).

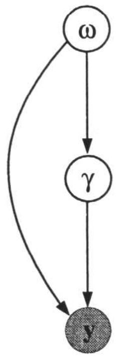

Figure 2: A graphical representation of a mixture of FA or PCA models.

As in FA, a vector of observed values is assumed to arise as a linear function of a continuous latent vector, and the components of the latent vector are assumed to be mutually independent. Whereas in FA the latent vector is assumed to be Gaussian, in ICA the latent vector is assumed to have a more general probability density—in particular, one for which the independence assumption has stronger consequences than mere decorrelation. In a nonlinear generalization of ICA (Lee, Lewicki, and Sejnowski 2000), mixtures of ICA can be used to both represent and classify data. Attias (chapter 10) proposes another generalization in which the latent density models are represented flexibly via mixture-of-Gaussian densities. Conditional on the choices of mixture components of these underlying densities, one has a FA model. The graphical model is again a three-layer directed model with a discrete node representing the mixture components in the top level and continuous nodes in the two lower levels. The exact inference algorithm for this model scales exponentially in the number of components of the latent vector. To handle large models, Attias (chapter 10) develops a variational approximation.

Finally, another line of research involving mixture models is the *mixture of experts* architecture (Jacobs et al. 1991), which is a conditional density model appropriate for supervised learning. In this model both the "mixing proportion," namely, $P(\omega)$, and the "mixing components," namely, $P(\mathbf{y}|\omega)$, are conditioned on the input vector $\mathbf{x}$. The mixture model thus takes the form $P(\mathbf{y}|\mathbf{x}) = \sum_{\omega} P(\omega|\mathbf{x})P(\mathbf{y}|\omega,\mathbf{x})$. The conditioning allows the input space to be partitioned adaptively into a set of regions (via the $P(\omega|\mathbf{x})$ term) in which different regression or classification surfaces are fit (the $P(\mathbf{y}|\omega,\mathbf{x})$ term).

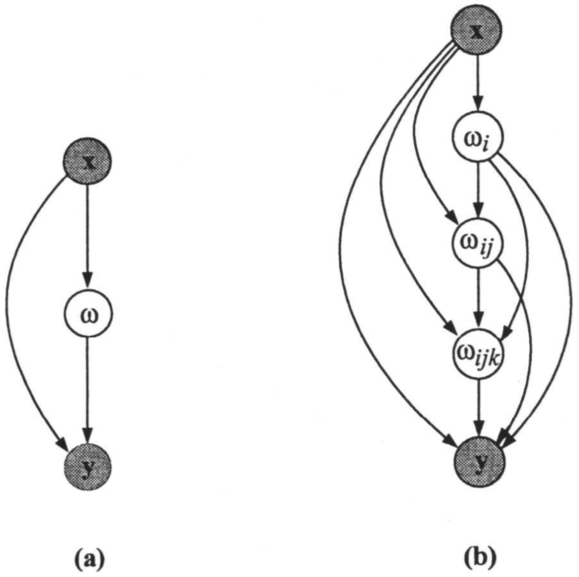

Figure 3: (a) The graphical representation of a mixture of experts model. This is a mixture model in which the distributions of the hidden variable ω and the output vector **y** are both conditioned on the input vector **x**. (b) The graphical model representation of a three-level hierarchical mixture of experts model. This model involves a sequence of hidden variables, ω_i, ω_{ij}, and ω_{ijk}, corresponding to a probabilistic, nested partition of the input space.

The mixture of experts is shown as a graphical model in figure 3(a). Here we see the conditional dependence of both the discrete latent variable and the observable **y** on the input vector. Note that both the input vector and the output vector are observed (shaded).

Jordan and Jacobs (chapter 11) generalized the mixture of experts to a hierarchical architecture (the *hierarchical mixture of experts*, or "HME"), in which a sequence of latent decisions are made, each of which is conditional on the input vector **x** and the previous decisions. The corresponding graphical model is shown in figure 3(b). Geometrically, the HME corresponds to a nested partitioning of the input space (the HME is essentially a probabilistic decision tree).

Jordan and Jacobs also proposed using the EM algorithm to fit the parameters of mixture-of-experts architectures. For the HME, the inner loop of this algorithm involves a recursive pass upward in the tree to compute the posterior probabilities of the latent decision nodes (conditioned on both **x** and **y**). As should be expected from figure 3(b), this

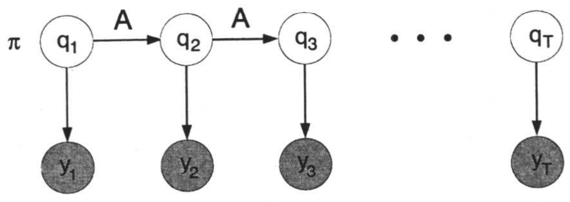

Figure 4: A hidden Markov model represented as a graphical model. Each horizontal slice corresponds to a time step and is isomorphic to the mixture model shown in figure 1. The distribution π is the *initial state distribution* and A is the *state transition matrix*. The *output sequence* $(y_0, y_1, \ldots, y_T)$ is observed and the *state sequence* $(q_0, q_1, \ldots, q_T)$ is unobserved.

recursion is a special case of the general inference algorithms for graphical models.

Dynamical Models

The hidden Markov model (HMM) is a paradigm example of a tractable graphical model. As shown in figure 4, the HMM can be viewed as a dynamical generalization of the basic mixture model in which the mixture model is copied and there are additional edges joining the hidden nodes. Each such node can be in one of M states, and there is an $M \times M$ transition matrix parameterizing these edges. The inference problem is that of calculating the probabilities of the hidden nodes given the entire sequence of observed nodes. This problem is solved via a recursive algorithm (the "alpha-beta algorithm") that proceeds forward and backward in the graph.

Smyth, Heckerman and Jordan (chapter 1) review the general graphical model formalism, describing in particular the *junction tree algorithm* for exact inference in graphical models. They then discuss HMMs, deriving the alpha-beta algorithm from the point of view of the junction tree framework. Several variants of the basic HMM architecture are also presented.

The technique of copying a basic underlying graphical model and linking nodes in the copies to obtain a Markovian dynamical model is widespread (Dean and Kanazawa 1989). Pursuing this approach in the case of factor analysis yields the classical linear-Gaussian Markov model, much studied in systems theory (cf. Roweis and Ghahramani 1999). The inference problem is solved by an analog of the forward-backward algorithm in which the "forward" algorithm is the classical Kalman filter. Both this forward recursion and any of a number of backward algo-