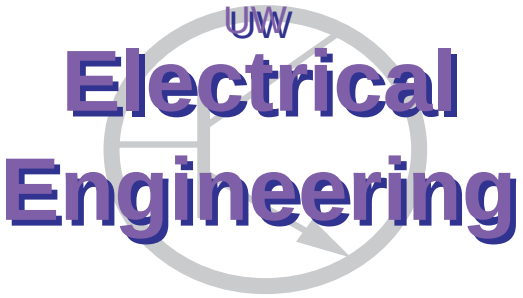
Graphical Models and Automatic Speech Recognition

Jeffrey A. Bilmes



UWEE Technical Report
Number UWEETR-2001-0005
November 2001

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Graphical Models and Automatic Speech Recognition

Jeffrey A. Bilmes

University of Washington, Dept. of EE, UWEETR-2001-0005

November 2001

Abstract

Graphical models provide a promising paradigm to study both existing and novel techniques for automatic speech recognition. This paper provides a general overview of graphical models and their uses as statistical models. It is shown that the underlying statistical assumptions behind many pattern recognition techniques commonly used for speech recognition can be described by a graph – this includes Gaussians distributions, mixture models, decision trees, factor analysis, principle component analysis, linear discriminant analysis, and hidden Markov models. Moreover, this paper shows that many advanced models proposed for speech recognition and language processing can also be described by a graph. A number of speech recognition techniques born directly out of the graphical-models paradigm are also surveyed. In conducting this survey, it becomes apparent that the space of models describable by a graph is enormous. As will be seen, it seems quite probable that a thorough exploration of the space of models easily describable by a graph would yield ultimately a model that performs far better than the HMM. Given the overview presented in this paper, it will be easier to begin such an endeavor.

1 Introduction

Since its inception, the field of automatic speech recognition (ASR) [124, 36, 20, 154, 78, 84, 112, 75] has increasingly come to rely on statistical methodology, moving away from approaches that were initially proposed such as template matching, dynamic time warping, and non-probabilistically motivated distortion measures. While there are still many successful instances of heuristically motivated techniques in ASR, it is becoming increasingly apparent that a statistical understanding of the speech process can only improve the performance of an ASR system. Perhaps the most famous example is that of the hidden Markov model [124], currently the predominant approach to ASR and a statistical generalization of earlier template-based practices.

A complete state-of-the-art ASR system involves numerous separate components, many of which are statistically motivated. Developing a thorough understanding of a complete ASR system, when it is seen as a collection of such conceptually distinct entities, can take some time. A impressive achievement would be an over-arching and unifying framework within which most statistical ASR methods can be accurately and succinctly described. Fortunately, a great many of the successful algorithms used by ASR systems can be described in terms of graphical models.

Graphical models (GMs) are a flexible statistical abstraction that have been successfully used to describe problems in a variety of domains ranging from medical diagnosis and decision theory to time series prediction and signal coding. Intuitively, GMs merge probability theory and graph theory. They generalize many techniques used in statistical analysis and signal processing such as Kalman filters[65], auto-regressive models [105], and many information-theoretic coding algorithms [48]. They provide a visual graphical language with which one may observe and reason about some of the most important properties of random processes, and the underlying physical phenomena these processes are meant to represent. They also provide set of computationally efficient probability computation and decision-making algorithms. Overall, GMs encompass an extremely large family of statistical techniques.

GMs provide an excellent formalism within which to study and understand ASR algorithms. With GMs, one may rapidly evaluate and understand a variety of different algorithms, since typically they have only minor graphical differences. As we will see in this paper, many of the existing statistical techniques in ASR appear to be representable using GMs — apparently no other known abstraction possesses this property. And even though the set of algorithms currently used in ASR is large, this collection occupies a relatively small volume within GM algorithm space. Because so many existing ASR successes lie within this under-explored space, it is likely that a systematic study of GM-based ASR algorithms could lead to new more successful approaches to ASR.

GMs can also help to reduce programmer time and effort. First, when described by a graph, it is easy to see if a statistical model appropriately represents relevant information contained in a corpus of (speech) data. GMs can help to rule out a statistical model which might otherwise require a large amount of programming effort to evaluate. A GM moreover can be minimally designed so that it has representational power only where needed. This means that a GM-based system might have smaller computational demands than a model designed without the data in mind, further easing programmer effort. Secondly, with the right set of computational tools, many considerably different statistical algorithms can be rapidly evaluated in a speech recognition system. This is because the same underlying graphical computing algorithms are applicable for all graphs, regardless of the algorithm represented by the graph. Section 5 briefly describes the new graphical models toolkit (GMTK)[12], which is one such tool that can be used for this purpose.

Overall, this paper argues that it is both pedagogically and scientifically useful to portray ASR algorithms in the umbrage of GMs. Section 1 provides an overview of GMs showing how they relate to standard statistical procedures. It also surveys a number of GM properties (Section 2.1), such as probabilistic inference and learning. Section 3 casts many of the methods commonly used for automatic speech recognition (ASR) systems as instances of GMs and their associated algorithms. This includes principle component analysis [40], linear discriminant analysis (and its quadratic and heteroschedastic generalizations) [97], factor analysis, independent component analysis, Gaussians densities, multi-layered perceptrons, mixture models, hidden Markov models, and many language models. This paper further argues that developing novel ASR techniques can benefit from a GM perspective. In doing so, it surveys some recent techniques in speech recognition, some of which have been developed without GMs explicitly in mind (Section 4), and some of which have (Section 5).

In this paper, capital letters will refer to random variables (such as X, Y and Q) and lower-case letters will refer to values they may take on. Sets of variables may be referred to as X_A or Q_B where A and B are sets of indices. Sets may be referred to using a Matlab-like range notation, such as $1:N$ which indicates all indices between 1 and N inclusive. Using this notation, one may refer to a length T vector of random variable taking on a vector of values as $P(X_{1:T} = x_{1:T})$.

2 Overview of Graphical Models

Broadly speaking, graphical models offer two primary features to those interested in working with statistical systems. On the one hand, a GM may be viewed as an abstract, formal, and visual language that can depict important properties (conditional independence) of natural systems and signals when described by multi-variate random processes. There are mathematically precise rules that describe what a given graph means, rules which associate with a graph a family of probability distributions. Natural signals (those which are not purely random) have significant statistical structure, and this can occur at multiple levels of granularity. Graphs can show anything from causal relations between high-level concepts [117] down to the fine-grained dependencies existing within the neural code [5]. On the other hand, along with GMs come a set of algorithms for efficiently performing probabilistic inference and decision making. Typically intractable, the GM inference procedures and their approximations exploit the inherent structure in a graph in a way that can significantly reduce computational and memory demands.

Simply put, graphical models describe conditional independence properties amongst collections of random variables. A given GM is identical to a list of conditional independence statements, and a graph represents

all distributions for which all these independence statements are true. A random variable X is conditionally independent of a different random variable Y given a third random variable Z under a given probability distribution $p(\cdot)$, if the following relation holds:

$$p(X = x, Y = y|Z = z) = p(X = x|Z = z)p(Y = y|Z = z)$$

for all x, y, and z. This is written $X \perp\!\!\!\perp Y|Z$ and it is said that "X is independent of Y given Z under $p(\cdot)$". This has the following intuitive interpretation: if one has knowledge of Z, then knowledge of Y does not change one's knowledge of X and vice versa. Conditional independence is different from unconditional (or marginal) independence. Therefore, neither $X \perp\!\!\!\perp Y$ implies $X \perp\!\!\!\perp Y|Z$ nor vice versa. Conditional independence is a powerful concept — using conditional independence, a statistical model can undergo enormous simplifications. Moreover, even though conditional independence might not hold for certain signals, making such assumptions might yield vast improvements because of computational, data-sparsity, or task-specific reasons (e.g., consider the hidden Markov model with assumptions which obviously do not hold for speech [10], but which nonetheless empirically appear to be somewhat benign, and at times even helpful as described in Section 3.9). Formal properties of conditional independence are described in [98, 117].

A GM [98, 32, 149, 117, 79] is a graph $\mathcal{G} = (V, E)$ where V is a set of vertices (also called nodes or random variables) and the set of edges E is a subset of the set $V \times V$. The graph describes an entire *family* of probability distributions over the variables V. A variable can either be scalar- or vector-valued, where in the latter case the vector variable implicitly corresponds to a sub-graphical model over the elements of the vector. The edges E, depending on the graph semantics (see below), specifies a set of conditional independence properties over the random variables. The properties specified by the GM are true for all members of its associated family.

Four items must be specified when using a graph to describe a particular probability distribution: the GM *semantics*, *structure*, *implementation*, and *parameterization*. The semantics and the structure of a GM are inherent to the graph itself, while the implementation and parameterization are implicit within the underlying model.

2.0.1 Semantics

There are many types of GMs, each one with differing semantics. The set of conditional independence assumptions specified by a particular GM, and therefore the family of probability distributions it represents, can be different depending on the type of GM. The semantics specifies a set of rules about what is or is not a valid graph and what set of distributions correspond to a given graph. Various types of GMs include directed models (or Bayesian networks) [117, 79],[1] undirected networks (or Markov random fields) [25], factor graphs [48, 96], chain graphs [98, 128] which are combinations of directed and undirected GMs, causal models [118], decomposable models, dependency networks [71], and many others. When the semantics of a graph change, the family of distributions it represents also changes, but overlap can exist between certain families.

A Bayesian network (BN) [117, 79, 70] is one type of GM where the graph edges are directed and acyclic. In a BN, edges point from parent to child nodes, and the graph implicitly spells out a factorization that is a simplification of the chain rule of probability, namely:

$$p(X_{1:N}) = \prod_i p(X_i|X_{1:i-1}) = \prod_i p(X_i|X_{\pi_i}).$$

The first equality is the probabilistic chain rule, and the second equality holds under a particular BN, where π_i designates node i's parents according to the BN. A Dynamic Bayesian Network (DBN) [35, 61, 51, 156] has exactly the same semantics as a BN, but is structured to have a sequence of clusters of connected vertices, where edges between clusters point only in the direction of increasing time. DBNs are particularly useful to describe time signals such as speech.

[1] Note that the name "Bayesian network" does not imply Bayesian statistical inference. In fact, both Bayesian and non-Bayesian Bayesian networks may exist.

Several equivalent schemata exist that formally define a BN's conditional independence relationships [98, 117, 79]. The idea of d-separation (or directed separation) is perhaps the most widely known: a set of variables A is conditionally independent of a set B given a set C if A is d-separated from B by C. D-separation holds if and only if *all* paths that connect any node in A and any other node in B have a node v with either: 1) the arrows along the path do not converge at v and $v \in C$, or 2) the arrows along the path **do** converge at v, and neither v nor any descendant of v is in C. From d-separation, one may "read off" a list of conditional independence statements from a graph. This set of probability distributions for which this list of statements is true is precisely the set of distributions represented by the graph. Graph properties equivalent to d-separation include the directed local Markov property [98] (a variable is conditionally independent of its non-descendants given its parents), and the Bayes-ball procedure [135] (shown in Figure 1).

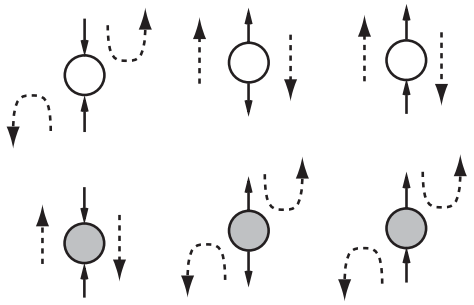


Figure 1: The Bayes-ball procedure makes it easy to answer questions about a given BN such as "is $X_A \perp\!\!\!\perp X_B | X_C$?", where X_A, X_B, and X_C are disjoint sets of nodes in a graph. The answer is true if and only if an imaginary ball bouncing from node to node along the edges in a graph and starting at any node in X_A can reach any in X_B. The ball must bounce according to the rules as depicted in the figure. Only the nodes in X_C are shaded. A ball may bounce through a node depending both on its shading and the direction of its edges. The dashed arrows depict whether a ball, when attempting to bounce through a given node, may bounce through that node or if it is blocked and must bounce back.

Conditional independence properties in undirected graphical models (UGMs) are much simpler than for BNs, and are specified using graph separation. For example, assuming that X_A, X_B, and X_C are disjoint sets of nodes in an UGM, $X_A \perp\!\!\!\perp X_B | X_C$ is true when all paths from any node in X_A to any node in X_B intersect some node in X_C. In a UGM, a distribution may be described as a factorization of potential functions over the cliques in the graph.

BNs and DGMs are not the same. Despite the fact that BNs have complicated semantics, they are useful for a variety of reasons. One is that BNs can have a causal interpretation, where if node A is a parent of B, A can be thought of as a cause of B. A second reason is that the family of distributions associated with BNs is not the same as the family associated with UGMs — there are some useful probability models, for example, that are concisely representable with BNs but which are not representable at all with UGMs (and vice versa). This issue will arise in Section 3.1 when discussing Gaussian densities. UGMs and BNs do have an overlap, however, and the family of distributions corresponding to this intersection is known as the decomposable models [98]. These models have important properties relating to efficient probabilistic inference and graph type (namely, triangulated graphs and the existence of a junction tree).

In general, a lack of an edge between two nodes does not imply that the nodes are independent. The nodes might be able to influence each other indirectly via an indirect path. Moreover, the existence of an edge between two nodes does *not* imply that the two nodes are necessarily dependent — the two nodes could still be independent for certain parameter values or under certain conditions (see later sections). A GM guarantees only that the lack of an edge implies some conditional independence property, determined according to the graph's semantics. It is therefore best, when discussing a given GM, to refer only to its (conditional) independence rather than its dependence properties.

Originally BNs were designed to represent causation, but more recently, models with semantics [118] more

precisely representing causality have been defined. Other directed graphical models have been designed as well [71], and can be thought of as the general family of directed graphical models (DGMs).

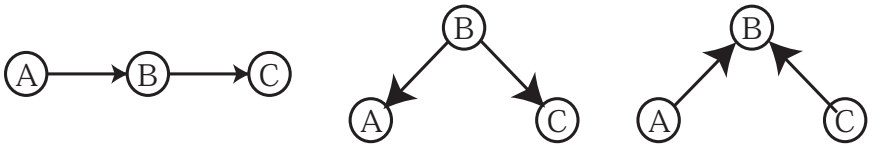


Figure 2: This figure shows four BNs with different arrow directions over the same random variables, A, B, and C. On the left side, the variables form a three-variable first-order Markov chain $A \rightarrow B \rightarrow C$. In the middle graph, the same conditional independence statement is realized even though one of the arrow directions has been reversed. Both these networks state that $A \perp\!\!\!\perp C|B$. These two networks do not, however, insist that A and B are not independent. The right network corresponds to the property $A \perp\!\!\!\perp C$ but it does not imply that $A \perp\!\!\!\perp C|B$. Perhaps show the a DGM not representable by a UGM and vice versa.

2.0.2 Structure

A graph's structure, the set of nodes and edges, determines the set of conditional independence properties for the graph under a given semantics. Note that more than one GM might correspond to exactly the same conditional independence properties even though their structure is entirely different. In this case, multiple very different looking graphs correspond to the same family of probability distributions. In such cases, the various GMs are said to be Markov equivalent [143, 144, 72]. In general, it is not immediately obvious with complicated graphs how to visually determine if Markov equivalence holds, but algorithms are available which can determine the members of an equivalence class [143, 144, 109, 28].

Nodes in a graphical model can be either *observed*, or *hidden*. If a variable is observed, it means that its value is known, or that data (or "evidence") is available for that variable. If a variable is hidden, it currently does not have a known value, and all that is available is the conditional distribution of the hidden variables given the observed variables (if any). Hidden nodes are also called confounding, latent, or unobserved variables. Hidden Markov models are so named because they possess a Markov chain that, in many applications, contains only hidden variables.

A node may switch roles, and may sometimes be hidden and at other times be observed. With an HMM, for example, the "hidden" chain might be observed during training (because a phonetic or state-level alignment has been provided) and hidden during recognition (because the hidden variable values are not known for test speech data). When making the query "is $A \perp\!\!\!\perp B|C$?", it is implicitly assumed that C is observed. A and B are the nodes being queried, and any other nodes in the network not listed in the query are considered hidden. Also, when a collection of sampled data exists (say as a training set), some of the data samples might have missing values each of which would correspond to a hidden variable. The EM algorithm [37], for example, can be used to train the parameters of hidden variables.

Hidden variables and their edges reflect a belief about the underlying generative process lying behind the phenomenon that is being statistically represented. This is because the data for these hidden variables is either unavailable, is too costly or impossible to obtain, or might even not exist since the hidden variables might only be hypothetical (e.g., specified by hand based on human-acquired knowledge about the underlying domain). Hidden variables can be used to indicate the underlying causes behind an information source. In speech, for example, hidden variables can be used to represent the phonetic or articulatory gestures, or more ambitiously, the originating semantic thought behind a speech waveform.

Certain GMs allow for what are called *switching* dependencies [60, 110, 15]. In this case, edges in a GM can change as a function of other variables in the network. An important advantage of switching dependencies is the reduction in the required number of parameters needed by the model. Switching dependencies are also used in a new graphical model-based toolkit for ASR [12] (see Section 5). A related construct allows GMs to have optimized local probability implementations [50].

It is sometimes the case that certain observed variables are only used as conditional variables. For example, consider the graph $B \rightarrow A$ which implies a factorization of the joint distribution $P(A, B) = P(A|B)P(B)$. In many cases, it is not necessary to represent the marginal distribution over B. In such cases B is a "conditional-only" variable, meaning is always and only to the right of the conditioning bar. In this case, the graph represents $P(A|B)$. This can be useful in a number of cases including classification (or discriminative modeling), where we might only be interested in posterior distributions over the class random variable, or in situations where additional observations, say Z, exist which might be marginally independent of a class variable, say C, but which, conditioned on other observations, say X, are dependent. This can be depicted by the graph $C \rightarrow X \leftarrow Z$, where it is assumed that the distribution over Z is not represented.

Often, the true (or the best) structure for a given task is unknown. This can mean that either some of the edges or nodes (which can be hidden) or both can be unknown. This has motivated research on learning the structure of the model from the data, with the general goal to produce a structure that accurately reflects the important statistical properties in the data set. These can take a Bayesian [70, 72] or frequentist point of view [23, 94, 70]. Structure learning is akin to both statistical model selection [102, 24] and data mining [34]. Several good reviews of structure learning are presented in [23, 94, 70]. Structure learning from a discriminative perspective, thereby producing what could be called *discriminative generative models*, was proposed in [10].

Figure 3 depicts a topological hierarchy of both the semantics and structure of GMs, and shows where different models fit in.

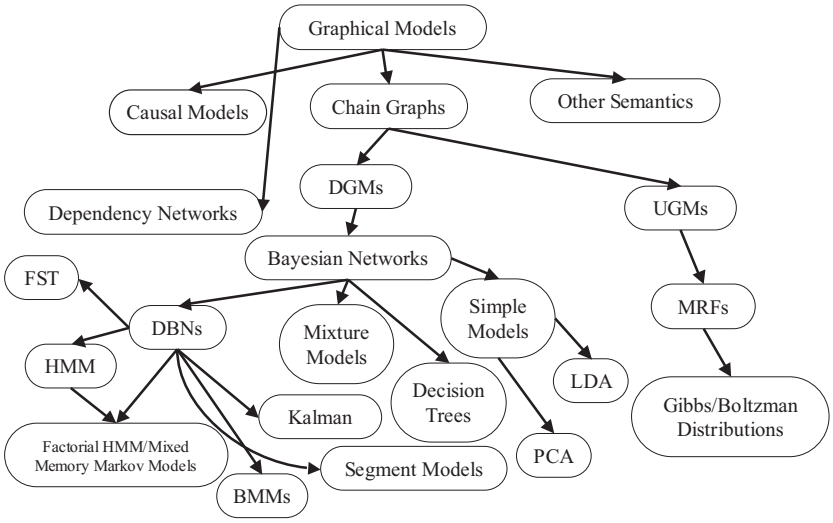


Figure 3: A topology of graphical model semantics and structure

2.0.3 Implementation

When two nodes are connected by a dependency edge, the local conditional probability representation of that dependency may be called its *implementation*. A dependence of a variable X on Y can occur in a number of ways depending on if the variables are discrete or continuous. For example, one might use discrete conditional probability tables (CPTs), compressed tables [50], decision trees, or even a deterministic function (in which case GMs may represent data-flow [1] graphs, or may represent channel coding algorithms [48]). A node in a GM can also depict a constant input parameter since random variables can themselves be constants. Alternatively, the dependence might be linear regression models, mixtures thereof, or non-linear regression (such as a multi-layered perceptron [18], or a STAR [139] or MARS [49] model). In general, different edges in a graph will have different implementations.

In UGMs, conditional distributions are not represented explicitly. Rather a joint distribution over all the nodes in the graph is specified with a product of what are called "potential" functions over cliques in the graph. In general the clique potentials could be anything, although particular types are commonly used (such as Gibbs or Boltzmann distributions [74]). Many such models fall under what are known as exponential models [40]. The implementation of a dependency in an UGM is implicitly specified via these functions in that they specify the way in which one variable can influence the resulting probabilities for other random variable values.

2.0.4 Parameterization

The parameterization of a model corresponds to the parameter values of a particular implementation in a particular structure. For example, with linear regression, parameters are simply the regression coefficients; for a discrete probability table the parameters are the table entries. Since parameters of distributions which are random can themselves be seen as nodes, Bayesian approaches may easily be represented [70] with GMs.

Many algorithms exist for training the parameters of a graphical model. These include maximum likelihood [40] such as the EM algorithm [37], discriminative or risk minimization approaches [140], gradient descent [18], sampling approaches [104], or general non-linear optimization [45]. The choice of algorithm depends both on the structure and implementation of the GM. For example, if there are no hidden variables, an EM approach is not required. Certain structural properties of the GM might render certain training procedures less crucial to the performance of the model [15]

2.1 Efficient Probabilistic Inference

A key application of any statistical model is to compute the probability of one subset of random variables given values for some other subset, a procedure known as probabilistic inference. Inference is essential both to make predictions based on the model and to learn the model parameters using, for example, the EM algorithm [37, 108]. One of the critical advantages of GMs is that they offer procedures for making exact inference as efficient as possible, much more so than if conditional independence is ignored or is used unwisely. And if the resulting savings is not enough, there are GM-inspired approximate but still more efficient inference algorithms.

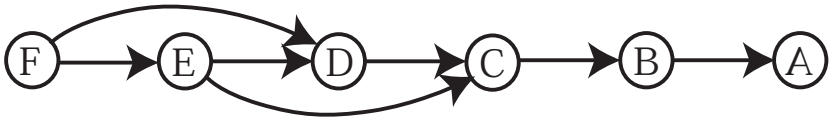


Figure 4: The graph's independence properties are used to move sums inside of factors.

Exact inference can in general be quite computationally costly. For example, suppose there is a joint distribution over 6 variables $p(a, b, c, d, e, f)$ and the goal is to compute $p(a|f)$. This requires both $p(a, f)$ and p(f), so the variables b, c, d, e must be "marginalized", or integrated away to form $p(a, f)$. The naive way of performing this computation would entail the following sum:

$$p(a, f) = \sum_{b,c,d,e} p(a, b, c, d, e, f)$$

Supposing that each variable has K possible values, this computation requires $O(K^6)$ operations, a quantity which is exponential in the number of variables in the joint distribution. If, on the other hand, it was possible to factor the joint distribution into factors containing fewer variables, it would be possible to reduce computation significantly. For example, under the graph in Figure 4, the above distribution may be factored as follows:

$$p(a, b, c, d, e, f) = p(a|b)p(b|c)p(c|d, e)p(d|e, f)p(e|f)p(f)$$

so that the sum

$$p(a,f) = p(f) \sum_b p(a|b) \sum_c p(b|c) \sum_d p(c|d,e) \sum_e p(d|e,f) p(e|f)$$

requires only $O(K^3)$ computation. Inference in GMs involves formally defined manipulations of graph data structures and then operations on those data structures. These operations provably correspond to valid operations on probability equations, and they reduce computation essentially by moving sums, as in the above, as far to the right as possible in these equations.

The graph operations and data structures needed for inference are typically described in their own light, without needing to refer back to the original probability equations. One well-known form of inference procedure, for example, is the junction tree (JT) algorithm [117, 79]. In fact, the commonly used forward-backward algorithm [124] for hidden Markov models is just a special case of the junction tree algorithm [136], and which is a special case of the generalized distributive law [2].

The JT algorithm requires that the original graph be converted into a junction tree, a tree of cliques with each clique containing nodes from the original graph. A junction tree possesses the running intersection property, where the intersection between any two cliques in the tree is contained in all cliques in the (necessarily) unique path between those two cliques. The junction tree algorithm itself can be viewed as a series of messages passing between the connected cliques of the junction tree. These messages ensure that the neighboring cliques are locally consistent (i.e., that the neighboring cliques have identical marginal distributions on those variables that they have in common). If the messages are passed in a particular order, called the message passing protocol, then because of the properties of the junction tree, local consistency guarantees global consistency, meaning that the marginal distributions on all common variables in all cliques are identical, meaning that inference is correct. Because only local operations are required in the procedure, inference can be fast.

For the junction tree algorithm to be valid, however, a decomposable model must first be formed from the original graph. Junction trees exist only for decomposable models, and a message passing algorithm can provably be shown to yield correct probabilistic inference only in that case. It is often the case, however, that a given DGM or UGM is not decomposable. In such case it is necessary to form a decomposable model from a general GM (directed or otherwise), and in doing so make fewer conditional independence assumptions. Inference is then solved for this larger family of models. Solving inference for a larger family still of course means that inference has been solved for the smaller family corresponding to the original non-decomposable model.

Two operations are needed to transform a general DGM into a decomposable model: moralization and triangulation. Moralization joins the unconnected parents of all nodes and then drops all edge directions. This procedure is valid because more edges means fewer conditional independence assumptions or a larger family of probability distributions. Moralization is required to ensure that the resulting UGM does not disobey any of the conditional independence assumptions made by the original DGM. In other words, after moralizing, it is assured that the UGM will make no independence assumption that is not maded by the original DGM.

After moralization, or if starting from a UGM to begin with, triangulation is necessary to produce a decomposable model. The set of all triangulated graphs corresponds exactly to the set of decomposable models. The triangulation operation [117, 98] adds edges until all cycles in the graph with non-consecutive nodes (along the cycle) have a connected pair. Triangulation is valid because more edges enlarge the set of distributions represented by the graph. Triangulation is necessary because only for triangulated (or decomposable) graphs do junction trees exists. A good survey of triangulation techniques is given in [93].

Finally, a junction tree is formed from the triangulated graph by, first, forming all maximum cliques in the graph, next connecting all of the cliques together into a "super" graph, and finally finding a maximum spanning tree [30] amongst that graph of maximum cliques. In this case, the weight of an edge between two cliques is set to the number of variables in the intersection of the two cliques.

For a discrete-node-only network, junction tree complexity is $O(\sum_{c \in C} \prod_{v \in c} |v|)$ where C is the set of cliques in the junction tree, c is the set of variables contained within a clique, and $|v|$ is the number of possible values of variable v. The algorithm is exponential in the clique sizes, a quantity important to

minimize during triangulation. There are many ways to triangulate [93], and unfortunately the operation of finding the optimal triangulation is itself NP-hard. For an HMM, the clique sizes are N^2, where N is the number of HMM states, and there are T cliques leading to the well known $O(TN^2)$ complexity for HMMs. Further information on the junction tree and related algorithms can be found in [79, 117, 32, 80].

Exact inference, such as the above, is useful only for moderately complex networks since inference is NP-hard in general [29]. Approximate inference procedures can, however, be used when exact inference is not feasible. There are several approximation methods including variational techniques [133, 76, 81], Monte Carlo sampling methods [104], and loopy belief propagation [146]. Even approximate inference can be NP-hard however [33]. Therefore, it is always important to use a minimal model, one with least possible complexity that still accurately represents the important aspects of a task.

3 Graphical Models and Automatic Speech Recognition

A wide variety of algorithms often used in state-of-the-art ASR systems can easily be described using GMs, and this section surveys a number of them. While many of these approaches were developed without GMs in mind, they turn out to have surprisingly simple and elucidating network structures. Given an understanding of GMs, it is in many cases easier to understand the technique by looking first at the network than at the original algorithmic description.

As is often done, the following sections will separate ASR algorithms into three categories: acoustic, pronunciation, and language modeling. Each of these are essentially statistical models about how the speech data that we observe is generated. Different statistical models, and inference within these models, leads us to the different techniques, but each are essentially special cases of the more general GM techniques described above.

Notation. In general x will be a N-dimensional observation vector, which we will often write as $x_{1:N}$. X is a random variable. x^i is the i^{th} sample of X.

3.1 The Acoustic Level: Gaussians

The most successful and widely used density for acoustic modeling in ASR systems is the multi-dimensional Gaussian. The Gaussian density has a deceptively simple mathematical description that does not disclose many of the useful properties this density possesses (such as that first and second moments are the natural and sufficient parameters of the distribution). In this section, it will be shown how Gaussians can be viewed as either undirected or directed GMs, and how each of these views can succinctly describe quite distinct properties of the Gaussian.

An N-dimensional Gaussian density has the form:

$$p(x) = p(x_{1:N}) = |2\pi\Sigma|^{-1/2} e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)}$$

where μ is an N-dimensional mean vector, and Σ is an $N \times N$ covariance matrix. Typically, $K = \Sigma^{-1}$ refers to the inverse covariance (or the concentration) matrix of the density. It will be useful to form partitions of a vector x into a number of parts. For example, a bi-partition of $x = [x_A \; x_B]$ may be formed, where x_A and x_B are sub-vectors of x [63], and where the sum of the dimensions of x_A and x_B equals N. Tri-partitions $x = [x_A \; x_B \; x_C]$ may also be formed. In this way, the mean vector $\mu = [\mu_A \; \mu_B]^T$, and the covariance and concentration matrices can be so partitioned as

$$\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix} \quad \text{and} \quad K = \begin{pmatrix} K_{AA} & K_{AB} \\ K_{BA} & K_{BB} \end{pmatrix}.$$

Conventionally, $\Sigma_{AA}^{-1} = (\Sigma_{AA})^{-1}$, so that the sub-matrix operator takes precedence over the matrix inversion operator. A well known property of Gaussians is that if $\Sigma_{AB} = 0$ then x_A and x_B are marginally independent ($x_A \perp\!\!\!\perp x_B$).

A more interesting property of a Gaussian is that for a given tri-partition $x = [x_A \; x_B \; x_C]$ of x, and corresponding tri-partitions of μ and K, then $x_A \perp\!\!\!\perp x_B | x_C$ if and only if, in the corresponding tri-partition of K, $K_{AB} = 0$.

This property may be proven quite readily. For any distribution, the chain rule of probability says

$$p(x) = p(x_A|x_B)p(x_B).$$

When $p(x)$ is a Gaussian density, the marginal distribution $p(x_B)$ is also Gaussian with mean μ_B and covariance Σ_{BB}. Furthermore, $p(x_A|x_B)$ is a Gaussian having a "conditional" mean and covariance [106, 4]. Specifically, the distribution for x_A given x_B is a conditional Gaussian with an x_B-dependent mean vector

$$\mu_{A|B} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B)$$

and a fixed covariance matrix

$$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}.$$

This means that, if the two vectors X_A and X_B are jointly Gaussian, then given knowledge of one vector, say X_B, the result is a Gaussian distribution over X_A which has a fixed variance for all values of x_B but has a mean which is a linear transformation of the particular value of x_B. Most importantly, it can be shown that K_{AA}, the upper-left partition of the original concentration matrix K, is the inverse of the conditional covariance, specifically $K_{AA} = \Sigma_{A|B}^{-1}$ [98, 149].

Let the partition X_A be further partitioned to form the sub-bi-partition $x_A = [x_{Aa} \; x_{Ab}]$, meaning that $x = [x_{Aa} \; x_{Ab} \; x_B]$. A similar sub-partition is formed of the concentration matrix

$$K_{AA} = \begin{pmatrix} K_{AAaa} & K_{AAab} \\ K_{AAba} & K_{AAbb} \end{pmatrix}.$$

Setting $K_{AAab} = 0$ implies that $x_{Aa} \perp\!\!\!\perp x_{Ab}$. Here, however, the independence holds only when conditioning on x_B. This yields the result desired, but with the matrix and vector partitions renamed. Therefore, zeros in the inverse covariance matrix result in conditional independence properties for a Gaussian, or more specifically if $K_{ij} = 0$ then $X_i \perp\!\!\!\perp X_j | X_{\{1:N\}\setminus\{i,j\}}$.

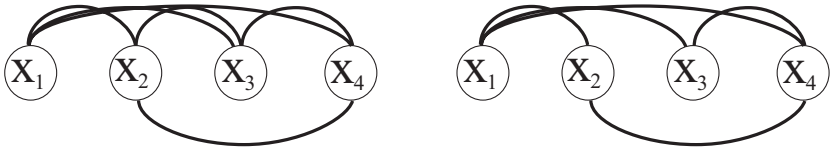


Figure 5: A Gaussian viewed as an UGM. On the left, there are no independence assumptions. On the right, $X_2 \perp\!\!\!\perp X_3 | \{X_1, X_4\}$.

This property of Gaussians corresponds to their view as an UGM. To see this, first consider a fully connected UGM with N nodes, which represents all Gaussians. Setting an entry, say K_{ij}, to zero corresponds to the independence property above, which corresponds in the UGM to removing an edge between variable x_i and x_j. This is shown in Figure 5. Therefore, missing edges in a Gaussian UGM correspond exactly to zeros in the inverse covariance matrix.

A Gaussian may also be viewed as a BN, and in fact many BNs. Unlike with a UGM, to form a Gaussian BN a specific variable ordering must first be chosen, the same ordering used to factor the joint distribution with the chain rule of probability. A Gaussian can be factored

$$p(x_{1:N}) = \prod_i p(x_i|x_{i+1:N})$$

according to some fixed but arbitrarily chosen variable ordering. Each factor is a Gaussian with conditional mean

$$\mu_{i|i+1:N} = \mu_i + \Sigma_{i,i+1:N}(\Sigma_{i+1:N,i+1:N})^{-1}(x_{i+1:N} - \mu_{i+1:N})$$

and conditional covariance

$$\Sigma_{i|i+1:N} = \Sigma_{ii} - \Sigma_{i,i+1:N}(\Sigma_{i+1:N,i+1:N})^{-1}\Sigma_{i+1:N,i}.$$

both of which are unique for a given ordering (these are an application of the conditional Gaussian formulas above, but with A and B set to the specific values i and $(i+1):N$ respectively). Therefore, the chain rule expansion can be written:

$$p(x_{1:N}) = \prod_i (2\pi\Sigma_{i|i+1:N})^{-1/2} e^{-\frac{1}{2}(x_i - \mu_{i|i+1:N})^2 \Sigma^{-1}_{i|i+1:N}} \tag{1}$$

An identical decomposition of this Gaussian can be produced in a different way. Every concentration matrix K has a unique factorization $K = U'DU$ where U is a unit upper-triangular matrix and D is diagonal [106, 68]. A unit triangular matrix is a triangular matrix that has ones on the diagonal, and so has a unity determinant (so is non-singular), therefore, $|K| = |D|$. This corresponds to a form of Cholesky factorization $K = R'R$, where R is upper triangular, $D^{1/2} = \text{diag}(R)$ is the diagonal portion of R, and $R = D^{1/2}U$. A Gaussian density can therefore be represented as:

$$p(x) = (2\pi)^{-d/2}|D|^{1/2} e^{-\frac{1}{2}(x-\mu)'U'DU(x-\mu)}$$

The unit triangular matrices, however, can be "brought" inside the squared linear terms by considering the argument within the exponential

$$\begin{aligned}
(x-\mu)'U'DU(x-\mu) &= (U(x-\mu))'D(U(x-\mu)) \\
&= (Ux - \tilde{\mu})'D(Ux - \tilde{\mu}) \\
&= ((I-B)x - \tilde{\mu})'D((I-B)x - \tilde{\mu}) \\
&= (x - Bx - \tilde{\mu})'D(x - Bx - \tilde{\mu})
\end{aligned}$$

where $U = I - B$, I is the identity matrix, B is an upper triangular matrix with zeros along the diagonal, and $\tilde{\mu} = U\mu$ is a new mean. Again, this transformation is unique for a given Gaussian and variable ordering. This process exchanges K for a diagonal matrix D, and produces a linear auto-regression of x onto itself, all while not changing the Gaussian normalization factor contained in D. Therefore, a full-covariance Gaussian can be represented as a conditional Gaussian with a regression on x itself, yielding the following:

$$p(x_{1:N}) = (2\pi)^{-d/2}|D|^{1/2} e^{-\frac{1}{2}(x - Bx - \bar{\mu})^T D(x - Bx - \bar{\mu})}.$$

In this form the Gaussian can be factored where the i^{th} factor uses only the i^{th} row of B:

$$p(x_{1:N}) = \prod_i (2\pi)^{-d/2} D_{ii}^{1/2} e^{-\frac{1}{2}(x_i - B_{i,i+1:N}x_{i+1:N} - \bar{\mu}_i)^2 D_{ii}} \tag{2}$$

When this is equated with Equation (1), and note is taken of the uniqueness of both transformations, it is the case that

$$B_{i,i+1:N} = \Sigma_{i,i+1:N}(\Sigma_{i+1:N,i+1:N})^{-1}.$$

and that $\tilde{\mu}_i = \mu_i - B_{i,i+1:N}\mu_{i+1:N}$. This implies that the regression coefficients within B are a simple function of the original covariance matrix. Since the quantities in the exponents are identical for each factor (which are each an appropriately normalized Gaussian), the variance terms D_{ii} must satisfy:

$$D_{ii} = \Sigma^{-1}_{i|i+1:N}$$

meaning that the D_{ii} values are conditional variances.

Using these equations we can now show how a Gaussian can be viewed as a BN. The directed local Markov property of BNs states that the joint distribution may be factorized as follows:

$$p(x_{1:N}) = \prod_i p(x_i|x_{\pi_i})$$

where $\pi_i \subseteq (i+1):N$ are parents of the variable x_i. When this is considered in terms of Equation (2), it implies that the non-zero entries of $B_{i,i+1:N}$ correspond to the set of parents of node i, and the zero entries correspond to missing edges. In other words (under a given variable ordering) the B matrix determines the conditional independence statements for a Gaussians when viewed as a DGM, namely $X_i \perp\!\!\!\perp X_{\{(i+1):N\}\setminus\pi_i}|X_{\pi_i}$ if and only if the entries $B_{i,\{(i+1):N\}\setminus\pi_i}$ are zero.

It is important to realize that these results depend on a particular ordering of the variables $X_{1:N}$. A different ordering might yield a different B matrix, possibly implying different independence statements (depending on if the graphs are Markov equivalent, see Section 2.0.2). Moreover, a sparse B matrix for one ordering might become dense for a different ordering, and zeros in B may or may not yield zeros in $K = (I-B)'D(I-B)$ or $\Sigma = K^{-1}$, and vice versa. This means that a full covariance Gaussian with $N(N+1)/2$ non-zero covariance parameters might actually employ fewer than $N(N+1)/2$ parameters, since it is in the directed domain where sparse patterns of independence occur. For example, consider a 4-dimensional Gaussian with a B matrix such that $B_{12} = B_{13} = B_{14} = B_{24} = B_{34} = 1$, and along with the other zero entries, $B_{23} = 0$. For this B matrix, neither the concentration nor covariance matrices have any zeros, although they are full rank and it is true that $X_2 \perp\!\!\!\perp X_3|X_4$. It must be that K possesses redundancy in some way. The opposite (i.e., zeros in K or Σ, and no sparsity in B) may occur as well.

The question then becomes what form of Gaussian should be used, a DGM or a UGM, and if a DGM, in what variable order. A common goal is to minimize the total number of free parameters. If this is the case, the Gaussian should be represented in a "natural" domain [10], where the least degree of parameter redundancy exists. Sparse matrices often provide the answer, assuming no additional cost exists to represent sparse matrices, since the sparse pattern itself might be considered a parameter needing to be represented. This was exploited in [16], where the natural directed Gaussian representation was solicited from data, and where a negligible penalty in WER performance was obtained with a factored sparse covariance matrix having significantly fewer parameters.

It was mentioned earlier that UGMs and DGMs represent different families of probability distributions. It is interesting to note that Gaussians are able to represent any of the dependency structures captured either in a DGM (via an appropriate order of the variables and zeros in the B matrix) or a UGM (with appropriately placed zeros in the concentration matrix K). Therefore, Gaussians, along with many other interesting and desirable theoretical properties, are quite general in terms of their ability to possess conditional independence relationships. On the other hand, it is important to realize that while all UGM or DGM dependency structures can be realized by a Gaussian, the implementations in each case are only linear and the random components are only univariate Gaussian. A much greater family of distributions, other than just a Gaussian, can be depicted by a UGM or DGM, as we begin to see in the next sections.

3.2 The Acoustic Level: PCA/FA/ICA

Our second example of GMs for speech consists of techniques commonly used to transform speech feature vectors prior to being used in ASR systems. These include principle component analysis (PCA), factor analysis (FA), and independent component analysis (ICA). Interestingly, PCA has exactly the same graph structure as FA and ICA – the only difference lies in the implementation of the dependencies.

Some graphs in this section show both random variables and their parameters. For example, if X is Gaussian with mean μ, a μ node might be present as a parent of X. For our purposes, these nodes constitute constant random variables whose probability score is not counted (they are conditional-only variables, essentially always to the right of the conditioning bar). In a more general Bayesian setting [72, 70], however, these nodes would be true random variables with their own distributions and hyper-parameters.

In PCA, observations of a random vector X are assumed to be Gaussian with mean μ and covariance Σ. The spectral decomposition theorem [137] says that $\Sigma = \Gamma\Lambda\Gamma^T$, where Γ is an orthonormal (rotation) matrix, and Λ is a diagonal matrix of variances. The variable Y, a decorrelated linear transformation of X, can be formed using $Y = \Gamma^T(X - \mu)$. This follows since $E[YY^T] = \Gamma^T E[(X-\mu)(X-\mu)^T]\Gamma = \Gamma^T\Sigma\Gamma = \Lambda$.

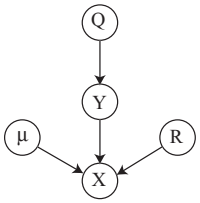


Figure 6: Principle components analysis (PCA) and factor analysis (FA) as a GM. The graph corresponds to the general equation $X = AY + \mathcal{N}(\mu, R)$, and $Y \sim \mathcal{N}(0, Q)$, where X is a random conditional Gaussian with mean $AY + \mu$ and variance R. X therefore has a linear dependency (with parameters A) on Y, and Y is itself a Gaussian with zero mean and Q variance. With PCA, Q is diagonal, and $R = 0$. With FA, $Q = I$, the identity matrix, and R is diagonal. Other generalizations are of course possible.

Solving for X yields the linear conditional "Gaussian":

$$X = \Gamma Y + \mu = \mathcal{N}(\Gamma Y + \mu, 0)$$

which says that X, conditioned on Y, is a constant conditionally-Gaussian random variable with mean $\Gamma Y + \mu$ and zero variance. All the variance in X is therefore explained by Y. Figure 6 shows this as a GM, where the dependency implementations are all linear. It is said that $Y \sim \mathcal{N}(0, \Lambda)$, a vector of independent Gaussian variables, are the "principle components" of X. Therefore, our model is that X is Gaussian, but more importantly that X is a transformation of some underlying hidden continuous "causes" Y which themselves are zero-mean diagonal-covariance Gaussians. PCA (or the KL-transform) is a transformation from the observations X to the most likely values of the hidden causes (i.e., the conditional mean $E[Y|X]$) since $p(y|x) = \mathcal{N}(\Gamma^T(x - \mu), \Lambda)$. It is often the case that only the largest principle components are used, in which case the dimensionality of Y is less than that of X.

There are many properties of PCA [106], a notable one (which motivates factor analysis below) is that PCA is not scale invariant — if the scale of X changes (say by converting from inches to centimeters), both Γ and Λ will also change, leading to different Y. In this sense, PCA explains the variance in X using only variances found in the hidden causes Y.

Factor analysis is only a simple modification of PCA. In this case,

$$X = \Gamma Y + \mathcal{N}(\mu, R) = \mathcal{N}(\Gamma Y + \mu, R)$$

where now $Y \sim \mathcal{N}(0, I)$, with I the identity matrix and R a non-negative diagonal matrix. In factor analysis, Γ is the factor loading matrix and Y the common factor vector. The residual term $U = X - \Gamma Y - \mu$, where $U \sim \mathcal{N}(0, R)$ are called the specific factors, and account for the underlying variance in X. In other words, X has its own variance, even conditional on Y — Y is unable to explain variance in X since Y is forced to have I as a covariance matrix. Γ, on the other hand, is forced to represent just the correlation between elements of X irrespective of its individual variance terms. As in PCA, it can be seen that in FA X is being explained by underlying hidden causes Y, and the same GM (Figure 6) explains FA.

In both PCA and FA, the hidden causes Y are marginally independent. Any dependencies between elements of the observations X exist only in how they are jointly dependent on the hidden causes Y. Another GM depicting this for PCA and FA is provided in Figure 7. More specifically, $p(y_j) = \mathcal{N}(y_j; 0, \lambda_j)$ and that $p(x_i) = \mathcal{N}(x_i; \sum_j \Gamma_{ij} y_j, r_j)$ where $r_j = 0$ for PCA and $\lambda_j = 1$ for FA.

Independent component analysis (ICA) [7, 87] is another method that explains data vectors X with hidden independent causes. Like PCA and FA, a goal of ICA is to first learn the parameters of the model

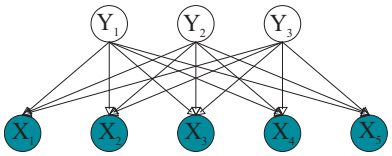


Figure 7: A graph showing the explicit dependencies for PCA, FA, and ICA. For PCA and FA, the dependencies are linear and Gaussian, for ICA the dependencies can be arbitrary, each one leading to a different ICA algorithm.

that explain X. Once done, it is possible to find the Y for a given X that consist of descriptions of X and that are statistically as independent as possible. Unlike PCA and FA, however, dependency implementations in ICA neither need to be linear nor Gaussian. Since the graph in Figure 7 does not depict implementations, the vector Y can be any non-linear and/or non-Gaussian causes of X. The graph insists only that the elements of Y are marginally independent, leaving alone the operations needed to compute $E[Y|X]$. Therefore, ICA can be seen simply as supplying the mechanism for different implementation of the dependencies used to infer $E[Y|X]$.

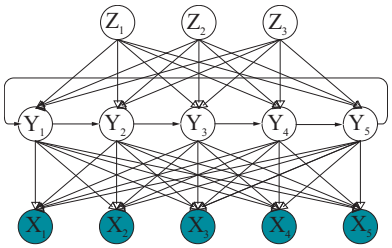


Figure 8: Multi-level ICA

.

Still other generalizations of PCA/FA/ICA can be obtained simply by using different implementations of the basic graph given in Figure 7. Independent factor analysis [6] occurs when the hidden causes Y are described by a mixture of Gaussians. Moreover, a multi-level factor analysis algorithm, shown in Figure 8, can be easily imagined where the middle hidden layer is a possibly non-independent explanation for the final marginally independent components. The goal once again is to train parameters to explain X, and to be able to compute $E[Z|X]$. As can be seen, using the graph it is easy to understand all of these techniques. Furthermore, simple changes in the graph structure and implementations of the dependencies can lead to dramatically different statistical procedures.

3.3 The Acoustic Level: LDA/QDA/MDA/QMDA

When the goal is pattern classification [40], it is often beneficial to first transform a set of features X to a space not spanned by the principle or independent components, but rather to where the features best discriminatively represent the classes. Let C be a variable that indicates the class of the current vector X, so that there are $|C|$ classes. As above, a linear transformation can be used to maximize the between-class covariance while minimizing the within-class covariance in the transformed space. Specifically, the goal is to find the matrix A, the linear transformation on the observations X to form $Y = AX$ that maximizes $\mathrm{tr}(BW^{-1})$ [52] where

$$W = \sum_i p(C = i) E_{p(y|C=i)}[(y - \mu_y^i)(y - \mu_y^i)^T]$$

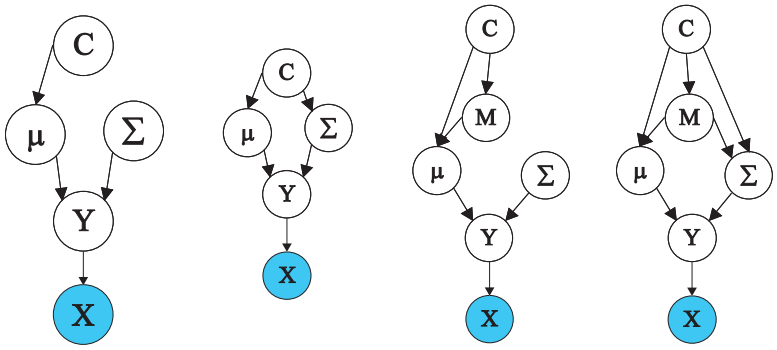


Figure 9: Linear discriminant analysis (left), and its generalizations.

and

$$B = \sum_i p(C = i)(\mu_y^i - \mu_y)(\mu_y^i - \mu_y)^T$$

where μ_y^i is the class conditional mean and μ_y is the global mean in the transformed space. This is a multi-dimensional generalization of Fisher's original linear discriminant analysis (LDA) formulation [44].

Like above, LDA is a particular statistical modeling assumption about the way in which observation samples X are generated. In particular, it is assumed that the class conditional distributions in the transformed space $P(Y|C = i)$ are Gaussians. Assuming priors $P(C = i)$, this means that Y is a mixture model $p(y) = \sum_i p(C = i)p(y|C = i)$, and classification of y is optimally performed using the posterior distribution:

$$p(C = i|y) = \frac{p(y|i)p(i)}{\sum_j p(y|j)p(j)}$$

For standard LDA, it is assumed that the Gaussian components $p(y|j) = \mathcal{N}(y; \mu_j, \Sigma)$ all have the same covariance matrix, and are distinguished only by the different means. The goal is to find the linear transformation from X to Y which maximizes the data likelihood. This suggests that the statistical model behind LDA can be graphically described as shown on the left in Figure 9.

The above two views of LDA can be intuitively unified as follows. Consider, for example, two class conditional distributions described by multi-dimensional Gaussians having identical covariance matrices. In this case, the discriminant functions are linear, and effectively project any unknown sample down to a linear subspace which points in the direction of the difference between the two means [40]. This linear subspace contains all of the discriminability between the two Gaussians. More generally, consider the subspace spanned by the means of C class-conditional identical covariance Gaussians. With distinct means and C classes, this subspace has rank $C - 1$. The discriminability is captured entirely within this subspace since the decision regions are hyperplanes orthogonal to the lines containing pairs of means [40]. The linear projection of X onto the $C - 1$ dimensional subspace Y spanned by the means leads to no loss in classification accuracy, assuming Y indeed is perfectly described with such a mixture. Note that the projected space has a maximum dimensionality of $C - 1$. If fewer than $C - 1$ dimensions are used for the projected space, this can lead to a dimensionality reduction algorithm that has a minimum loss in discriminatory information. Interestingly, it can be shown [97] that the transform above from the original formulation of LDA is identical to the maximum likelihood linear transformation from the observations X to Y under the model assumption given in Figure 9.

Viewed as a graphical model, it is relatively easy to extend LDA to more general techniques. The simplest extension allows for different covariance matrices so that $p(x|i) = \mathcal{N}(x; \mu_i, \Sigma_i)$, leading to the second GM in Figure 9. This has been called quadratic discriminant analysis (QDA) [40, 108], because decision boundaries are quadratic rather than linear, or heteroscedastic discriminant analysis (HDA) [97], because covariances are not identical. In the latter case, it is assumed that only a portion of the mean vectors and covariance

matrices are class specific – the portion that isn't corresponds in the projected space to the dimensions that do not carry discriminative information. Further generalizations are immediate. For example, if the class conditional distributions are Gaussians mixtures, every component sharing the same covariance matrix, then mixture discriminant analysis (MDA) [69] is obtained (2nd from right in Figure 9). A further generalization yields what could be called be called heteroscedastic MDA, as described on the far right in Figure 9. If non-linear dependencies are allowed between the hidden causes and the observed variables, then one may obtain non-linear discriminant analysis methods, similar to the neural-network feature preprocessing techniques [46, 90, 73] which have recently been used.

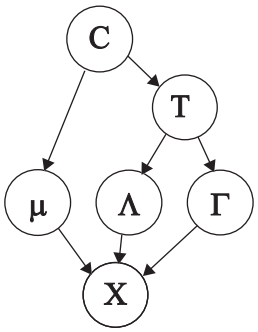


Figure 10: Semi-tied covariance matrices as GMs.

Taking note of the various factorizations one may perform on a positive-definite matrix [68], a concentration matrix K within a Gaussian distribution can be factored as $K = \Lambda'\Gamma\Lambda$. Using such a factorization, each Gaussian component in a Gaussian mixture can use one each from a shared pool of Λs and Γs, leading to what are called semi-tied covariance matrices [57, 155]. Once again, this form of tying can be described by a GM as shown in Figure 10.

3.4 Mixture Models

In speech recognition, hidden Markov model observation distributions rarely use only single component Gaussian distributions. Much more commonly, mixtures of such Gaussians are used. A general mixture distribution for $p(x)$ assumes the existence of a hidden variable C which determines which of each mixture component is active as in:

$$p(x) = \sum_i p(x, C = i) = \sum_i p(C = i)p(x|C = i)$$

where $p(x|C = i)$ is a component of the mixture. A GM may simply describe a general mixture distribution as shown in the graph $C \rightarrow X$. Conditional mixture generalizations, where X requires Z, are quite easy to obtain using the graph $Z \rightarrow C \rightarrow X$, leading to the equation:

$$p(x|z) = \sum_i p(x, C = i, z) = \sum_i p(C = i|z)p(x|C = i)$$

Many texts exist describe the various properties of mixture distributions such as [138, 107], some of which can be described in this way.

3.5 The Acoustic Level: Acoustic Classifier Combination

It has often been found that when multiple separately trained classifiers are used to make a classification decision in tandem, the resulting classification error rate often decreases. This has been found in many

instances both empirically and theoretically [77, 95, 119, 150, 18]. The theoretical results often make assumptions about the statistical dependencies amongst of the various classifiers, such as that their errors are assumed to be statistically independent. The empirical results for ASR have found that combination is useful at the acoustic feature level [11, 89, 67, 90], the HMM state level [91], the sub-word or word level [153], and even at the utterance level [43].

Assume that $p_i(c|x)$ is a probability distribution corresponding to the i^{th} classifier, where c is the class for feature set x. A number of classification combination rules exist such as the sum rule [92] where $p(c|x) = \sum_i p_i(c|x)$, or the product rule where $p(c|x) \propto \prod_i p_i(c|x)$. Each of these schemes can be explained statistically, by assuming a statistical model which validly leads to the particular combination rule. Ideally, the combination rule which performs best will correspond to the model which best matches the data. For example, the sum rule corresponds to a mixture model described above, and the product rule can be derived by the independent assumptions corresponding to a naive Bayes classifier [17]. Additional combination schemes, moreover, can be defined under the assumption of different models, some of which might not require the errors to be statistically independent.

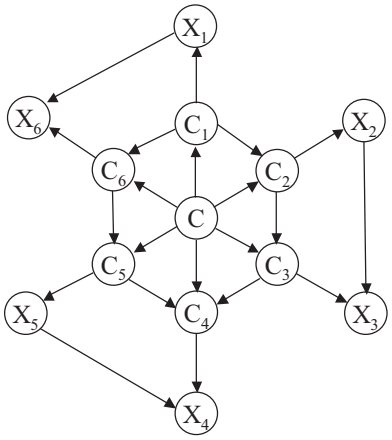


Figure 11: General GM to describe classifier combination. The model does not require in all cases that the errors are statistically independent.

More advanced combination schemes can be defined by, of course, assuming more sophisticated models. One such example is shown in Figure 11, where a true class variable C drives several error-full (noisy) versions of the class C_i, each of which generates a (possibly quite dependent) set of feature vectors. By viewing the process of classifier combination as a graph, and by choosing the right graph, one may quickly derive combination schemes that best matches the data available and that need not make assumptions which might not be true.

3.6 The Acoustic Level: Adaptation

It is typically the case that additional ASR WER improvements can be obtained by additional adaptation of the model parameters after training has occurred, but before the final utterance hypothesis is decided upon. Broadly, these take the form of vocal-tract length normalization (VTLN) [86], and explicit parameter adaptation such as maximum-likelihood linear regression (MLLR) [99]. It turns out that these procedures may also be described with a GM.

VTLN corresponds to augmenting an HMM model with an additional global discrete hidden variable which indicates the vocal tract length. This variable indicates which transformation on the acoustic feature vectors should be performed. It is common in VTLN that all such transformations are attempted, and the one that yields the highest likelihood scores according to a model is chosen as the transformation. The graph in Figure 12 shows this model, where A is the variable which indicates vocal tract length, and which could

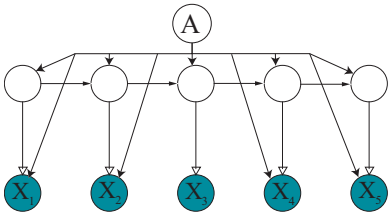


Figure 12: General GM to describe various adaptation and global parameter transformation methods, such as VTLN, MLLR, and SAT. The variable A indicates that the parameters of the entire model can be adapted.

affect the distribution for the rest of the model (in this case, the figure shows a hidden Markov model, to be described in a later section). If the procedure above corresponds to a Viterbi inference approach, then only the most likely assignment of A is used, and A can be a conditional-only variable (see Section 2.0.2). If a distribution is assumed over A, it is also possible to integrate over all values of A to produce the final probability score.

MLLR [99] corresponds to additionally training all the parameters of a model for a specific new speaker that was not necessarily encountered in the training set. Typically, the first set of answers that are hypothesized for an unknown utterance are used as the true answers (unsupervised adaptation) and the acoustics corresponding to that utterance along with that hypothesis are used as additional training material. A global transformation (or a small set of several global transformations) is used to update the model parameters which are then used to generate a new hypothesis, and the entire process might repeat. This procedure can also be described in Figure 12, where A in this case indicates the global parameter transformation.

Finally, speaker adaptive training (SAT) [3] involves simultaneously learning both the underlying speaker-unspecific parameters of a model along with speaker-specific transformations that are applicable for individual speakers. The transformation is an attempt to map from speaker-specific acoustic parameter space to a speaker independent space, which is where the remaining unspecific acoustic parameters are learned. This procedure corresponds to a statistical model where a speaker variable exists and is observed during training and is hidden during testing. The speaker variable is the parent of the variables used to produce a probability score, so Figure 12 once again describes the basic structure. The speaker variable determines which mapping is used for the acoustic mapping. During recognition, either the most likely transformation can be used (a Viterbi approach), or all speaker transformations can be attempted to produce an integrative score.

In all of these cases, different implementations of the edges in Figure 12 between A and the rest of the model can be used, and this would lead to novel forms of VTLN, MLLR, or SAT.

3.7 The Pronunciation Level

Pronunciation modeling in ASR systems involves examining each word in a lexicon, and finding sets of phone strings each of which describes a valid instance of the corresponding word [26, 129, 41, 47, 84]. Often these strings are specified probabilistically, where the probability of a given phone depends on preceding phone (as in a Markov chain) which is then used to produce the probability of the entire string (i.e., the pronunciation variant of a particular word). Sometimes the pronunciation also depends on the acoustics [47].

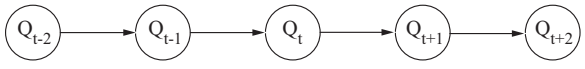


Figure 13: A simple first order Markov chain. This graph encodes the relationship $Q_t \perp\!\!\!\perp Q_{1:t-2} | Q_{t-1}$.

Using the chain rule, the probability of a string of T phone $V_{1:T}$, where V_i is a phone, can be written as:

$$P(V_{1:T}) = \prod_i p(V_t | V_{1:t-1}).$$

If it is assumed that only the previous K phone are relevant for determining the current phone probability, this yields a K^{th}-order Markov chain. Typically, only a first-order model is used for pronunciation modeling, as is depicted in Figure 13.

Phones are typically shared across multiple words. For example, in the two words "bat" and "bag", the middle phone /ae/ is the same. Therefore, it is advantageous in the acoustic Gaussian model for /ae/ to be shared between these two words. With a first-order model, however, it is possible only to select the distribution over the next state given the current one. This seems to present a problem since $P(V_t|/ae/)$ should choose a /t/ for "bat" and a /g/ for "bag". Clearly, then, there must be a mechanism, even in a first order case, to specify that the following V_t might need to depend on more than just the current phone.

Fortunately, there are several ways of achieving this issue. The easiest way is to expand the cardinality of V_t (i.e., increase the state space in the Markov chain). That is, the set of values of V_t corresponds not only to the set of different phone, but also to each of the phones in different positions of different words. Different values of V_{t-1}, corresponding to the same phone in different words, would then correspond to the same acoustic Gaussians, but the distribution of V_t given V_{t-1} could match whatever word V_{t-1} corresponds to. This procedure is equivalent to turning a K^{th}-order Markov chain into a first-order chain [78].

Another way to achieve such an effect is to, in addition to the previous phone, also condition on the word W_t and the sequential position in the word S_t as in $P(V_t|W_t, S_t)$. The word position is needed in case the same phone appears more than once with different following contexts in a single word. A Markov chain is then produced jointly over the two variables W_t and S_t. This approach is similar to expanding the graph in Figure 13 to explicitly mention the variables needed to keep track of and use each phone. This is described further in Section 5.

A GM view of a pronunciation model does not explicitly state the non-zero entries in the stochastic matrices in the Markov chain. Stochastic finite state automata (SFSA) [125] diagrams are better suited for that purpose. A GM, rather, explains only the independence structure of a model. It is important to realize that while SFSAs are often described using graphs (circles and arrows), SFSA graphs describe totally different properties of a Markov chain than do the graphs which are studied in this text.

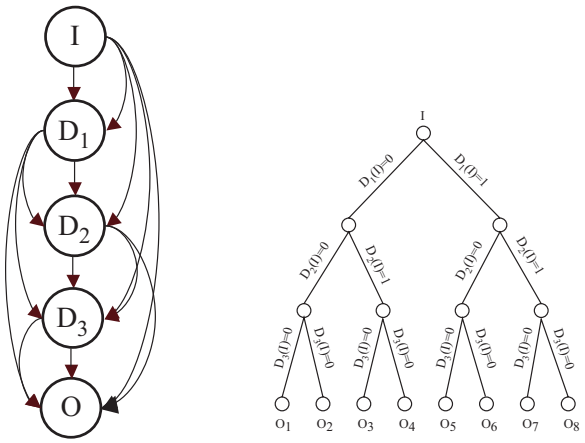


Figure 14: Left: A GM view of a decision tree, which is a probabilistic generalization of the more familiar decision tree on the right.

Pronunciation modeling often involves a mapping from base-forms (isolated word dictionary-based pronunciations) to surface forms (context-dependent and data-derived pronunciations). Decision trees are often used for this purpose [26, 129], so it is elucidative at this point to see how they may be described using BNs [81][2]. Figure 14 shows a decision tree on the right, and a stochastic version of the same decision tree as a GM on the left. In the graphical view, there is an input node I an output node O and a series of decision

[2] These GMs also describe hierarchical mixtures of experts [82].

nodes D_i. The cardinality of the decision variables correspond to the arity of the tree at the corresponding decision tree depth (which in the figure is two in each case).

In a GM-DT, a decision is made with a particular probability — all later questions which may be asked with non-zero probability are asked leading to a probability distribution over the final decisions O_i. For example, $D_1 = i$ corresponds to the i^{th} decision being made at the first level of the tree. This decision at this level occurs with probability $P(D_1 = i|I)$. The second decision is made with probability $P(D_2 = j|D_1, I)$ based on the first decision.

In a normal DT, only one decision is made at each level in the tree. We can recover such a "crisp" tree by insisting that the distributions at each level D_ℓ (and the final decision O) of the tree are Dirac-delta functions, such as $P(D_\ell = i|I) = \delta_{i,f_\ell(I,D_{1:\ell-1})}$ where $f_\ell(I, D_{1:\ell-1})$ is a deterministic function of the input I and previously made decisions $D_{1:\ell-1}$. Therefore, with the appropriate implementation of dependences, it can be seen that the final output O is a probabilistic generalization of normal DTs.

3.8 The Language Level

Similar to pronunciation modeling, the goal of language modeling is to produce a valid probability for all possible word strings $W_{1:T}$ in a language. There are many varieties of language models (LMs) [78, 130, 113, 27], and it is beyond the scope of this paper to describe them all. Nevertheless, the following section uses GMs to portray some of the most commonly and successfully used LMs.

Like pronunciation modeling, starting from a distribution over a word string the chain rule is applied and dependencies are dropped to get a K^{th} order Markov chain $p(W_t|H_t)$ where W_t is the current word and H_t is the previous K words. This leads to a graph identical to the one in Figure 13. Frustratingly, tri-grams (i.e., 2nd-order Markov chains over words) are the models that have at this point been the most universally used for language modeling [27].

We have already seen DGMs for Markov chains in the previous section. These graphs do not portray how the parameters of such models are obtained, a procedure which can be quite sophisticated [27]. In fact, much research on these models involves methods to cope with data-sparsity problems. Because of a lack of sufficient training data, smoothing methodology must be employed, whereby a K^{th} order model is mixed together with lower order models. The mixing weights are often determined by the frequency of occurrence of the word and history in training data [78, 27].

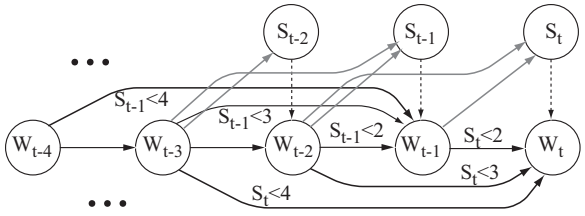


Figure 15: A GM view of a LM. The dashed arcs indicate that the parents are switching. The switching parents switch between a zeroth, first, and second order Markov chain. The switching parents might also posses previous words as parents, as shown by the grey arcs.

In the simplest of cases, a language model can be formed by smoothing a higher order Markov chain together with lower order ones, as in the following:

$$\begin{aligned} p(w_t|w_{t-1}, w_{t-2}) &= \alpha_3(w_{t-1}, w_{t-2}) f(w_t|w_{t-1}, w_{t-2}) \\ &+ \alpha_2(w_{t-1}, w_{t-2}) f(w_t|w_{t-1}) \\ &+ \alpha_1(w_{t-1}, w_{t-2}) f(w_t) \end{aligned}$$

where $\sum_i \alpha_i = 1$ for all word histories, and where the α coefficients are some history dependent mixing values which determine when and where to use which model. Figure 15 shows this mixture using a switching BN. The variables S_t correspond to the α coefficients, and the edges marked with S_t being a particular value exist only in the case that S_t has those values. The dashed edges between S_t and W_t indicate that the S_t variables are switching rather than normal parents. The graph describes the statistical underpinnings of many commonly used techniques such as deleted interpolation [78], which is a form of parameter training for the S_t variables. Of course, much of the success of a language model depends on the form of smoothing that is used [27], and such methods are not depicted by a GM.

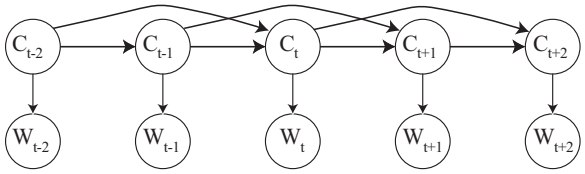


Figure 16: A class-based language model. Here, a Markov chain is used to model the dynamics of word classes rather than the words themselves.

A common extension to the above LM is to cluster words together and then form a Markov chain over the word group clusters, generally called a class-based LM [22]. There are a number of ways that these clusters can be formed, such as by grammatical category or by data-driven approaches which might use decision trees (as discussed in [78, 22]). Whatever the method, the underlying statistical model can also be described by a GM, as shown in Figure 16. This figure corresponds to the equation:

$$p(w_t|w_{t-1}, w_{t-2}) = \sum_{c_t} p(w_t|c_t)p(c_t|w_{t-1}, w_{t-2})$$

in which the Markov-chain exists over the (presumably) much lower dimensional class variables c rather than the high-dimensional word variables. This representation can therefore considerably decrease model complexity and therefore parameter training quality.

Note that many other language models are members of the family of exponential models[40]. These include those models whose parameters are learned by maximum entropy methods [121, 78, 9, 131], and are derived by establishing a number of constraints that the underlying probability distribution must possess. The goal is to find a distribution satisfying these constraints and otherwise having maximum entropy. Note that such an approach can be used to describe the distribution over an entire sentence[132], rather than a conditional distribution of the current word w_t given the current history h_t. In general, such maximum entropy models can be described by UGMs, where the edges between words indicate that there is some dependency induced by the constraint functions. In many cases, the resulting graphs can become quite complicated.

Overall, it is clear that there are a multitude of ways to depict language models with GMs, and this section has only begun to touch upon this topic, one which deserves at least an entire paper devoted to its exposition.

3.9 GMs for basic speech models

The Hidden Markov model (HMM) is still the most successful statistical technique used in ASR. The HMM encompasses standard acoustic, pronunciation, and most language modeling into a single unified framework. This is because pronunciation and language modeling can be seen as a large finite-state automata which can be "flattened" down to a single first-order Markov chain [111, 78]. This Markov chain consists of a sequence of serially connected discrete hidden variables during recognition, thus the name HMM.

Most generally, a hidden Markov model (HMM) is collection of T discrete scalar random variables $Q_{1:T}$ and T other variables $X_{1:T}$ which may be either discrete or continuous (and either scalar- or vector-valued).

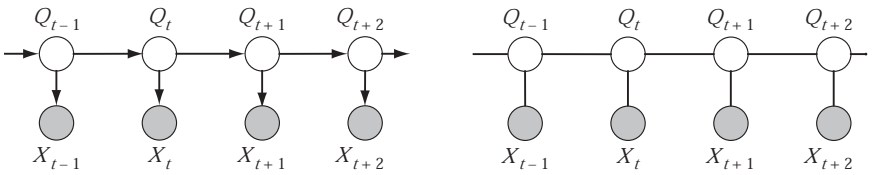


Figure 17: A hidden Markov model (HMM), viewed as a graphical model. Note that an HMM may be equivalently viewed either as a directed (left) or an undirected (right) model, in this case the conditional independence properties are the same.

These variables, collectively, possess the following conditional independence properties:

$$Q_{t:T} \perp\!\!\!\perp Q_{1:t-2} | Q_{t-1} \tag{3}$$

and

$$X_t \perp\!\!\!\perp \{Q_{\neg t}, X_{\neg t}\} | Q_t \tag{4}$$

for each $t \in 1 : T$. $Q_{\neg t}$ refers to all variables Q_τ except for the one at time $\tau = t$. The above set of conditional independence properties are minimal. If any are removed, the model is no longer an HMM. Furthermore, the properties are complete in that they are sufficient to prove any other conditional independence properties of an HMM. In other words, no other conditional independence properties are true in general of an HMM, unless they are derivable from Equations 3 and 4. The length T of these sequences is itself an integer-valued random variable having a complex distribution. An HMM consists of a hidden Markov chain of random variables (the unshaded nodes) and a collection of nodes corresponding to the speech utterance (the shaded nodes). In most ASR systems, the hidden chain corresponds to sequences of words, phones, and sub-phones.

This set of properties can be concisely described using the BN shown in Figure 17. The figure shows two equivalent representations of an HMM, one as a BN and another as an UGM. They are equivalent because moralizing the BN introduces no edges, and because the moralized HMM graph is already triangulated and therefore decomposable. The UGM on the right is the result of moralizing the BN on the left. Interestingly, the same graph describes the structure of a Kalman filter [65], in which case all the variables are continuous and Gaussian, and all dependency implementations are linear. Kalman filter operations are simply applications of the formulas for conditional Gaussians (Section 3.1), used in order to infer conditional means and covariances (the sufficient statistics for Gaussians).

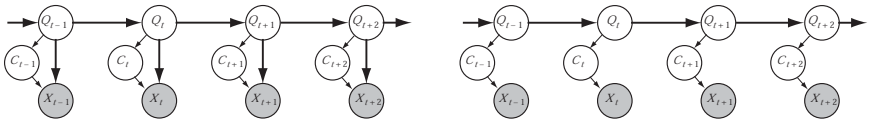


Figure 18: An HMM with mixture observation distributions (left) and a semi-continuous HMM (right).

In a standard HMM with Gaussian mixture observation densities, each value of the hidden variable (i.e., each state) corresponds to a separate (possibly tied) mixture distribution (Figure 18). Other forms of HMM also exist, such as when there is a single global pool of Gaussians, and each state corresponds to a particular mixture over this global pool. This is often called a semi-continuous HMM (similar to vector quantization [64]), and corresponds to the state-conditional observation equation:

$$p(x|Q = q) = \sum_i p(C = i|Q = q)p(x|C = i)$$

In other words, each state uses only a mixture amongst this globally shared set of components distributions. The GM for such an HMM looses an edge between Q and X as shown on the right in Figure 18. In this case, all of the represented dependence occurs via the hidden mixture variable at each time.

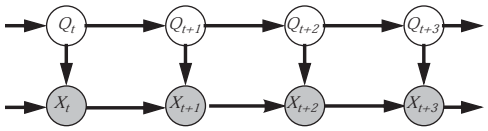


Figure 19: An Auto-regressive HMM as a GM

Still another modification of HMMs relaxes one of the HMM conditional independence statements, namely that successive feature vectors are conditionally independent given the state. Auto-regressive, or correlation HMMs [147, 21, 115], place an additional edges between successive observation vectors. In other words, the variable X_t might have as a parent not only the variable Q_t but also the variables X_{t-l} for $l = 1, 2, \ldots, K$ for some K. The case where $K = 2$ is shown in Figure 19. When the additional dependencies linear and Gaussian, these are sometimes called conditional Gaussian HMMs [115].

Note that although these models are sometimes called vector-valued auto-regressive HMMs, they are not to be confused with auto-regressive, linear predictive, or hidden filter HMMs [122, 123, 83, 124]. These latter models are HMMs which have been inspired from the use of linear-predictive coefficients for speech [124]. They use the observation distribution that arises from random Gaussian noise sources passed through a hidden-state dependent auto-regressive filter. The filtering occurs at the raw acoustic (signal) level rather than on the observation feature vector (frame) level. These earlier models can also be described by an GM which would describe the state-conditioned auto-regressive model at the speech sample level.

State-of-the-art ASR systems augment HMM feature vectors X_t with approximations to their first and second order time-derivatives (called delta- and delta-delta- features [42, 53, 54, 55], or just "dynamic" features). Most often, estimates of the derivative are obtained using linear regression [124], namely:

$$\dot{x}_t = \frac{\sum_{k=-K}^{K} k x_t}{\sum_{k=-K}^{K} k^2}$$

where K in this case is the number of points used to fit the regression. This can be viewed a regression because

$$\dot{x}_t = \sum_{k=-K}^{K} a_k x_{t-k} + \epsilon$$

where a_k are defined accordingly, and ϵ can be seen as a Gaussian error term. A new feature vector is then produced which consists of x_t and $\dot{x}_t$ appended together.

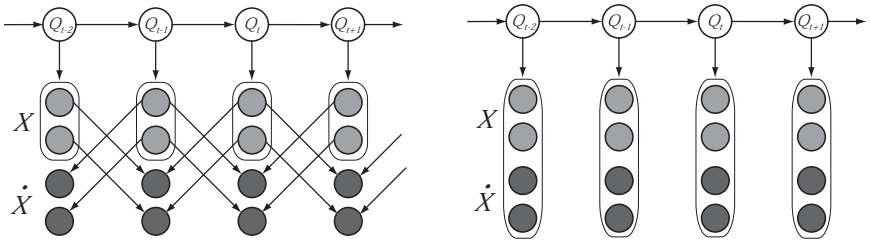


Figure 20: An GM-based explanation of why delta features work in HMM-based ASR systems. The left figure gives a GM that shows the generative process of HMMs with delta features. The right figure shows how delta features are typically used in an HMM system, where the information between $\dot{X}_t$ and Q_t is greatly increased relative to the left figure.

It is elucidating to expand the joint distribution of the features and the deltas, namely $p(x_{1:T}, \dot{x}_{1:T}) = \sum_{q_{1:T}} p(x_{1:T}, \dot{x}_{1:T}|q_{1:T})p(q_{1:T})$. The state conditioned joint distribution within the sum can be expanded as:

$$p(x_{1:T}, \dot{x}_{1:T}|q_{1:T}) = p(\dot{x}_{1:T}|x_{1:T}, q_{1:T})p(x_{1:T}|q_{1:T}).$$

The conditional distribution $p(x_{1:T}|q_{1:T})$ can be expanded as is normal for an HMM [124], but

$$p(\dot{x}_{1:T}|x_{1:T}, q_{1:T}) = \prod_t p(\dot{x}_t|\text{parents}(\dot{x}_t)).$$

This last equation follows because, observing the process to generate delta features, $\dot{X}_t$ is independent of everything else given its parents. The parents of $\dot{X}_t$ are a subset of $X_{1:T}$ and they do not include the hidden variables Q_t. This leads to the GM on the left in Figure 20, a generative model for HMMs augmented with delta features. Note that the edges between the feature stream X_t, and the delta feature stream $\dot{X}_t$ correspond to deterministic linear implementations. In this view, delta-features appear to be similar to fixed-dependency auto-regressive HMMs (Figure 19), where each child feature has additional parents both from the past and from the future. In this figure, however, there are no edges between $\dot{X}_t$ and Q_t, because $\dot{X}_t \perp\!\!\!\perp Q_t|\text{parents}(\dot{X}_t)$. This means that parents$(\dot{X}_t)$ contain all the information about $\dot{X}_t$, and Q_t is irrelevant.

It is often asked why delta features help ASR performance as much as they do. The left of Figure 20 does not portray the model typically used with delta features. A goal of speech recognition is for the features to contain as much information as possible about the underlying word sequence as represented via the vector $Q_{1:T}$. When the edges between $\dot{X}_t$ and its parents are removed, the mutual information [31] between $\dot{X}_t$ and Q_t can only *increase* (from zero to something greater). The right of Figure 20 thus shows the standard model used with deltas, where it is not the case that $\dot{X}_t \perp\!\!\!\perp Q_t$. Since in the right model, it is the case that more information about $\dot{X}_t$ and Q_t exist, it might be said that this model is inherently more discriminative (see Section 5).

Interestingly, the above analysis demonstrates that additional conditional independence assumptions (i.e., fewer edges) in a model can increase the amount of mutual information that exists between random variables. When edges are added between the delta features and the generative parents X_t, the delta features become less useful since there is less (or zero) mutual information between them and Q_t. Interestingly, the very conditional independence assumptions which are commonly seen as a flaw of the HMM appear to provide a benefit when using delta features. Can the removal of edges lead to and overall increase in the information between the joint random vector $X_{1:T}$ and $Q_{1:T}$? The data processing inequality [31] says it can not, but each feature vector, $(\dot{X}_t, X_t)$ will have more information about Q_t which can sometimes lead to better word error scores. This same analysis can be used to better understand other feature processing strategies derived from multiple frames of speech, such as LDA preprocessing over multiple windows [66] and other non-linear generalizations [46, 90, 73].

It has often been found that conditionally Gaussian HMMs (as in Figure 19) often do not provide an improvement when delta features are included in the feature stream [19, 21, 88, 148]. The above provides one possible explanation, namely that by having a delta feature $\dot{X}_t$ include as its parent say X_{t-1}, the mutual information between $\dot{X}_t$ and Q_t decreases (perhaps to zero). Note, however, that improvements were reported with the use of delta features in [151, 152] where discriminative output distributions were used. In [100, 101], successful results were obtained using delta features but where the conditional mean, rather than being linear, was non-linear and was implemented using a neural network. Furthermore, Buried Markov models [15] (to be described below) also found an improvement with delta features and additional dependencies, but only when the dependencies were added discriminatively.

Our last example of an augmented HMM is something often called an input-output HMM [8] (See Figure 21). In this case, there are variables at each time frame corresponding both to the input and the output. The output variables are to be inferred. Given a complete input feature stream $X_{1:T}$, one might want to find $E[Y|X]$, the most likely values for the output. These HMMs can therefore be used to map from a continuous variable length input feature streams to output stream. Such a model shows promise for speech enhancement.

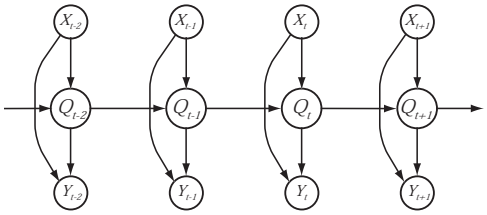


Figure 21: An input-output HMM. $X_{1:T}$ the input is transformed via integration over a Markov chain $Q_{1:T}$ into the output $Y_{1:T}$.

While HMMs account for much of the technology behind existing ASR, GMs include a much larger space of models. It seems quite improbable that within this space, it is the HMM alone that is somehow intrinsically superior to all other models. While there are of course no guarantees to the following, it seems reasonable to assume that because the space of GMs is large and diverse, and because it includes HMMs, that there exists some model within this space that will greatly outperform the HMM. The next section begins to explore more advanced speech models as viewed from a GM perspective.

4 GMs for advanced speech models

Many non-HMM models for speech have been developed outside the GM paradigm but turn out to be describable fairly easily as GMs — this section begins to describe some of them. While each of these models are quite different from each other, they can all be described with only simple modifications of some underlying graph structure.

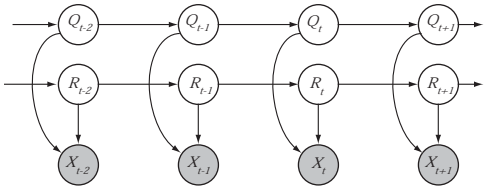


Figure 22: A factorial HMM where there are multiple hidden Markov chains.

The first example presented is a factorial HMM[62]. In this case, rather than a single Markov chain, multiple Markov chains are used to guide the temporal evolution of the probabilities over observation distributions (see Figure 22). The multiple hidden chains can be used to represent a number of real-world phenomena. For example, one chain might represent speech and another could represent an independent and dynamic noise source [85]. Alternatively, one chain could represent the speech to be recognized and the other chain could represent confounding background speech [141, 142] [3], or the two chains might each represent two underlying concurrent and independent sub-processes governing the realization of the observation vectors [56, 145, 103]. Such factored hidden state representations have also been called HMM decomposition [141, 142] in the past.

One can imagine many modifications of this basic structure, where dependencies are added between variables at each time step. Often, these separate Markov chains have been used for modeling separate loosely coupled streams of hidden articulatory information [126, 127] or to represent a coupling between phonetic and articulatory information [156].

It is interesting to note that the factorial HMMs described above are all special cases of HMMs. That is, they are HMMs with tied parameters and state transition restrictions made according to the factorization.

[3] A related method to estimate the parameters of a composite HMM given a collection of separate, independent, and already trained HMMs is called parallel model combination [59]

Starting with a factorial HMM consisting of two hidden chains Q_t and R_t, an equivalent HMM may be constructed by using $|\mathcal{Q}||\mathcal{R}|$ states and by restricting the set of state transitions and parameter assignments to be those only allowed by the factorial model. A factorial HMM using M hidden Markov chains each with K states that all span T time steps will have time complexity $O(TMK^{M+1})$ [62]. If one translates the factorial HMM into an HMM having K^M states, the complexity becomes $O(TK^{2M})$ which is significantly larger. An unrestricted HMM with K^M states will, however, have more expressive power than a factorial HMM with M chains each with K states because in the HMM there are no required state transition restrictions and any form of correlation may be represented between the separate chains. It is possible, however, that such an expanded state space would be more flexible than needed for a given task. Consider, as an example, the fact that many HMMs used for ASR have only simple left-to-right Markov chain structures.

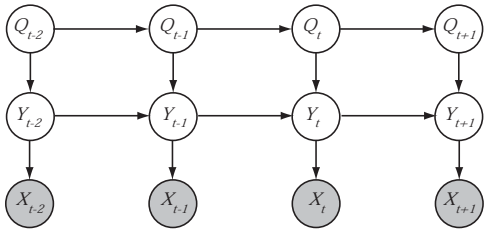


Figure 23: The GM corresponding to a switching Kalman filter (SKM). The Q variables are discrete, but the Y and X variables are continuous. In the standard SKM, the implementations between continuous variables are linear Gaussian, other implementations can be used as well and have been applied to the ASR problem.

As mentioned earlier, the GM for an HMM is identical to that of a Kalman filter — it is only the nodes and the dependency implementations that differ. Adding a discrete hidden Markov chain to a Kalman filter allows it to behave in much more complex ways than just a large joint Gaussian. This has been called a switching Kalman filter, as shown in Figure 23. A version of this structure, applied to ASR, has been called a hidden dynamic model [120]. In this case, the implementations of the dependences are such that the variables are non-linearly related.

Another class of models well beyond the boundaries of HMMs are called segment or trajectory models [115]. In such case, the underlying hidden Markov chain governs the evolution not of the statistics of individual observation vectors. Instead, the Markov chain determines the allowable sequence of observation segments, where each segment may be described using an arbitrary distribution. Specifically, a segment model uses the joint distribution over a variable length segment of observations conditioned on the hidden state for that segment. In the most general form, the joint distribution for a segment model is as follows:

$$\begin{aligned} & p(X_{1:T} = x_{1:T}) \\ &= \sum_{\tau} \sum_{q_{1:\tau}} \sum_{\ell_{1:\tau}} \prod_{i=1}^{\tau} p(x_{t(i,1)}, x_{t(i,2)}, \ldots, x_{t(i,\ell_i)}, \ell_i | q_i, \tau) p(q_i | q_{i-1}, \tau) p(\tau) \end{aligned} \tag{5}$$

There are T time frames and τ segments where the i^{th} segment is of a hypothesized length ℓ_i. The collection of lengths are constrained such that $\sum_{i=1}^{\tau} \ell_i = T$. For a particular segmentation and set of lengths, the i^{th} segment starts at time frame $t(i,1) = f(q_{1:\tau}, \ell_{1:\tau}, i, 1)$ and ends at time frame $t(i,\ell_i) = f(q_{1:\tau}, \ell_{1:\tau}, i, \ell_i)$. In this general case, the time variable t could be a general function $f()$ of the complete Markov chain assignment $q_{1:\tau}$, the complete set of currently hypothesized segment lengths $\ell_{1:\tau}$, the segment number i, and the frame position within that segment 1 through ℓ_i. It is assumed that $f(q_{1:\tau}, \ell_{1:\tau}, i, \ell_i) = f(q_{1:\tau}, \ell_{1:\tau}, i+1, 1) - 1$ for all values of all quantities.

Renumbering the time sequence for a segment starting at one, an observation segment distribution is given by:

$$p(x_1, x_2, \ldots, x_\ell, \ell | q) = p(x_1, x_2, \ldots, x_\ell | \ell, q) p(\ell | q)$$

where $p(x_1, x_2, \ldots, x_\ell|\ell, q)$ is the length ℓ segment distribution under hidden Markov state q, and $p(\ell|q)$ is the explicit duration model for state q.

A plain HMM may be represented using this framework if $p(\ell|q)$ is a geometric distribution in ℓ and if

$$p(x_1, x_2, \ldots, x_\ell|\ell, q) = \prod_{j=1}^{\ell} p(x_j|q)$$

for a state specific distribution $p(x|q)$. The stochastic segment model [116] is a generalization which allows observations in a segment to be additionally dependent on a region within a segment

$$p(x_1, x_2, \ldots, x_\ell|\ell, q) = \prod_{j=1}^{\ell} p(x_j|r_j, q)$$

where r_j is one of a set of fixed regions within the segment. A more general model is called a segmental hidden Markov model [58]

$$p(x_1, x_2, \ldots, x_\ell|\ell, q) = \int p(\mu|q) \prod_{j=1}^{\ell} p(x_j|\mu, q) d\mu$$

where μ is the multi-dimensional conditional mean of the segment and where the resulting distribution is obtained by integrating over all possible state-conditioned means in a Bayesian setting. More general still, in trended hidden Markov models [38, 39], the mean trajectory within a segment is described by a polynomial function over time. Equation 5 generalizes many models including the conditional Gaussian methods discussed above. A summary of segment models, their learning equations, and a complete bibliography is given in [115].

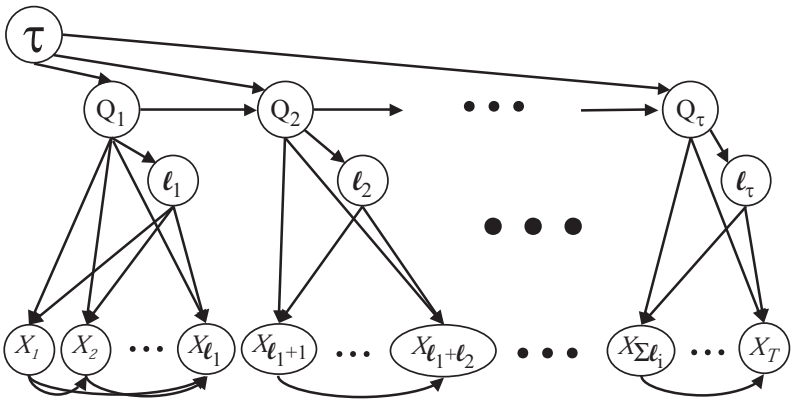


Figure 24: A Segment model viewed as a GM.

One can view a segment model as a GM, as shown in Figure 24. As shown in the graph, there is a single hidden variable τ which determines the number of segments, and within each segment additional dependencies exist. The segment model allows for the set of dependencies within a segment to be arbitrary, so it is likely that many of the dependencies shown in the figure would not exist in practice. Moreover, there may be additional dependencies not shown in the figure, since it is the case that there must be constraints on the segment lengths. Nevertheless, this figure quickly details the essential structure behind a segment model.

5 GM-motivated speech recognition

There have been several cases where graphical models have themselves been used as the cruxes of speech recognition systems — this section begins to explore several of them.

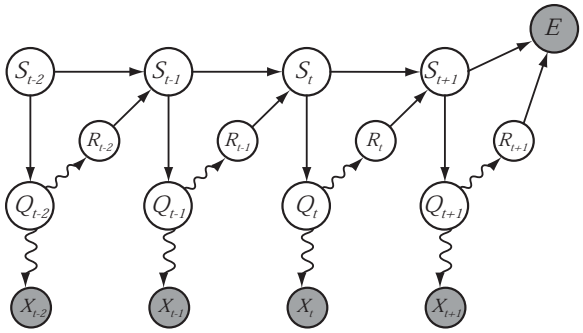


Figure 25: A BN used to explicitly represent parameter tying. In this figure, the straight edges correspond to deterministic implementations and the rippled edges correspond to stochastic implementations.

In Section 3.7 it was mentioned that for an HMM to keep track of the differences that exist between a phone that occurs in multiple contexts, it must expand the state space so that multiple HMM states share the same acoustic Gaussian mixture corresponding to a particular phone. It turns out that a directed graph itself may be used to keep track of the necessary parameter tying and to control the sequencing needed in this case [156]. The simplest of cases is shown in Figure 25, which shows a sequence of connected triangles — for each time frame a sequence variable S_t, a phone variable Q_t, and a transition variable R_t is used. The observation variable X_t has as its parent only Q_t since it is only the phone which determines the observation distribution. The other variables are used together to appropriately sequence through valid phones for a given utterance.

In this particular figure, straight lines are used to indicate that the implementations of the dependencies are strictly deterministic, and rippled lines are used to indicate that the implementations correspond to true random dependencies. This means, for example, that $p(S_{t+1} = i|R_t, S_t) = \delta_{i,f(R_t,S_t)}$ is a Dirac-delta function having unity probability for only one possible value of S_{t+1} given a particular pair of values for R_t and S_t.

In the figure, S_t is the current sequence number (i.e., 1, 2, 3, etc.) and indicates the sub-word position in a word (e.g., the first, second, or third phone). S_t does not determine the identity of the phone. Often, S_t will be a monotonically increasing sequence of successive integers, where either $S_{t+1} = S_t$ (the value stays the same) or $S_{t+1} = S_t + 1$ (an increment occurs). An increment occurs only if $R_t = 1$. R_t is a binary indicator variable which has unity value only when a transition between successive phone positions occurs. R_t is a true random variable and depending on the phone (Q_t), R_t will have a different binary distribution, thereby yielding the normal geometric duration distributions found in HMMs. Q_t is a deterministic function of the position S_t. A particular word might use a phone multiple times (consider the phone /aa/ in the word "yamaha"). The variable S_t sequences, say, from 1 through to 6 (the number of phones in "yamaha"), and Q_t then gets the identity of the phone via a deterministic mapping from S_t to Q_t for each position in the word (e.g., 1 maps to /y/, 2 maps to /aa/, 3 maps to /m/, and so on). This general approach can be extended to multiple hidden Markov chains, and to continuous speech recognition to provide graph structures that explicitly represent the control structures needed for an ASR system [156, 157, 12].

As mentioned above, factorial HMMs require a large expansion of the state space and therefore a large number of parameters. A recently proposed system that can model dependencies in a factorial HMM using many fewer parameters are called mixed memory markov models [134]. Viewed as a GM as in Figure 26, this model uses an additional hidden variable for each time frame and chain. Each normal hidden variables

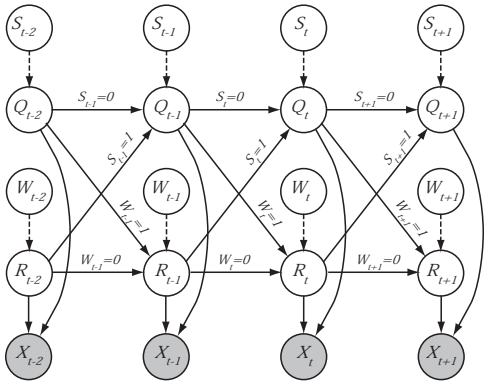


Figure 26: A mixed-memory hidden Markov model. The dashed edges indicate that the S and the W nodes are switching parents.

possesses an additional switching parent (as depicted by dotted edges in the figure, and as described in Section 2.0.2). The switching conditional independence assumptions for one time slice are that $Q_t \perp\!\!\!\perp R_{t-1} | S_t = 0$, $Q_t \perp\!\!\!\perp Q_{t-1} | S_t = 1$ and the symmetric relations for R_t. This leads to the following distributional simplification:

$$\begin{aligned} p(Q_t|Q_{t-1}, R_{t-1}) &= p(Q_t|Q_{t-1}, S_t = 0)P(S_t = 0) \\ &+ p(Q_t|R_{t-1}, S_t = 1)P(S_t = 1) \end{aligned}$$

which means that, rather than needing a single three-dimensional table for the dependencies, only two two-dimensional tables are required. These models have been used for ASR in [114].

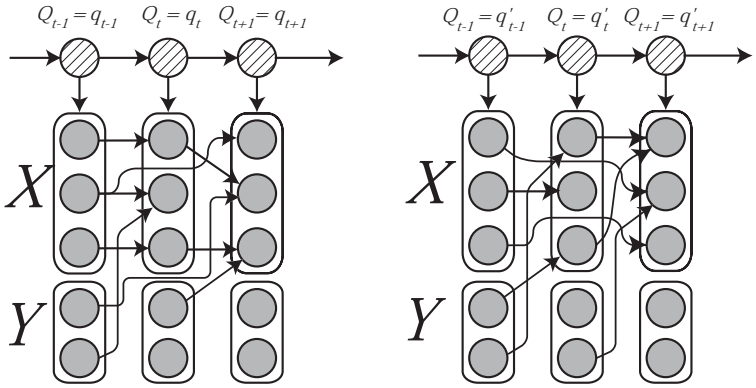


Figure 27: A Buried Markov Model (BMM) with two hidden Markov chain assignments, $Q_{1:T} = q_{1:T}$ on the left, and $Q_{1:T} = q'_{1:T}$ on the right.

A Buried Markov model (BMM) [15, 14, 13] is another recently proposed GM-based approach to speech recognition. A BMM is based on the idea that one can quantitatively measure where the conditional independence properties of a particular HMM are poorly representing a corpus of data. Wherever the model is found to be most lacking, additional edges are added (i.e., conditional independence properties are removed) relative to the original HMM. The BMM is formed to include only those data-derived, sparse, hidden-variable specific, and discriminative dependencies (between observation vectors) that are most lacking in the original model. In general, the degree to which $X_{t-1} \perp\!\!\!\perp X_t | Q_t$ is true can be measured using conditional mutual information $I(X_{t-1}; X_t|Q_t)$ [31]. If this quantity is zero, the model needs no extension, but if it is greater than zero, there is a modeling inaccuracy. Ideally, however, edges should be added discriminatively, to produce

a discriminative generative model, and when the structure is formed discriminatively, the notion has been termed structural discriminability [15]. For this purpose, the "EAR" (explaining away residual) measure has been defined that measures the discriminative mutual information between a variable X and its potential set of parents Z as follows:

$$\mathrm{EAR}(X, Z) \triangleq I(X; Z|Q) - I(X; Z)$$

It can be shown that choosing Z to optimize the EAR measure is equivalent to optimizing the posterior probability of the class Q [15]. Since it attempts to minimally correct only those measured deficiencies in a particular HMM, and since it does so discriminatively, this approach has the potential to produce better performing and more parsimonious models for speech recognition.

It seems apparent at this point that the set of models that can be described using a graph is enormous. With the options that are available in choosing hidden variables, the different sets of dependencies between those hidden variables, the dependencies between observations, choosing switching dependencies, and considering the variety of different possible implementations of those dependencies and the various learning techniques, it is obvious that the space of possible models is practically unlimited. Moreover, each of these modeling possibilities, if seen outside of the GM paradigm, requires a large software development effort before evaluation is possible with a large ASR system. This effort must be spent without having any guarantees as to the model's success.

In answer to these issues, a new flexible GM-based software toolkit has been developed (GMTK) [12]. GMTK is a graphical models toolkit that has been optimized for ASR and other time-series processing tasks. It supports EM and GEM parameter training, sparse linear and non-linear dependencies between observations, arbitrary parameter sharing, Gaussian vanishing and splitting, decision-tree implementations of dependencies, sampling, switching parent functionality, exact and log-space inference, multi-rate and multi-stream processing, and a textual graph programming language. The toolkit supports structural discriminability and arbitrary model selection, and makes it much easier to begin to experiment with GM-based ASR systems.

6 Conclusion

This paper has provided an introductory survey of graphical models, and then has provided a number of examples of how many existing ASR techniques can be viewed as instances of GMs. It is hoped that this paper will help to fuel the use of GMs for further speech recognition research. While the number of ASR models described in this document is large, it is of course the case that many existing ASR techniques have not even been given a mention. Nevertheless, it is apparent that ASR collectively occupies a relatively minor portion of the space of models representable by a graph. It therefore seems quite improbable that a thorough exploration of the space of graphical models would not ultimately yield a model that performs better than the HMM. The search for such a novel model should ideally occur on multiple fronts: on the one hand guided by our high-level domain knowledge about speech and thereby utilize phonetics, linguistics, morphology, and so on. On the other hand, the data should have a strong say, so there should be significant data-driven model selection procedures to determine the appropriate natural graph structure [10]. And since ASR is inherently an instance of pattern classification, the notion of discriminability (parameter training) and structural discriminability (structure learning) might play a key role in this search. All in all, graphical models opens many doors to novel speech recognition research.

References

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, Inc., Reading, Mass., 1986.

[2] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions in Information Thoery*, 46:325–343, March 2000.

[3] T. Anastasakos, J. McDonough, R. Schwartz, and J Makhoul. A compact model for speaker adaptive training. In *In Proceedings ICSLP*, pages 1137–1140, 1996.

[4] T.W. Anderson. *An Introduction to Multivariate Statistical Analysis.* Wiley Series in Probability and Statistics, 1974.

[5] J.J. Atick. Could information theory provide an ecological theory of sensory processing? *Network*, 3, 1992.

[6] H. Attias. Independent Factor Analysis. *Neural Computation*, 11(4):803–851, 1999.

[7] A.J. Bell and T.J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[8] Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1999.

[9] A. L. Berger, S.A. Della Pietra, and V.J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[10] J. Bilmes. *Natural Statistical Models for Automatic Speech Recognition.* PhD thesis, U.C. Berkeley, Dept. of EECS, CS Division, 1999.

[11] J. Bilmes, N. Morgan, S.-L. Wu, and H. Bourlard. Stochastic perceptual speech models with durational dependence. *Intl. Conference on Spoken Language Processing*, November 1996.

[12] J. Bilmes and G. Zweig. The graphical models toolkit: An open source software system for speech and time-series processing (submitted). *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2002.

[13] J.A. Bilmes. Data-driven extensions to HMM statistical dependencies. In *Proc. ICSLP*, Sidney, Australia, December 1998.

[14] J.A. Bilmes. Buried Markov models for speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, March 1999.

[15] J.A. Bilmes. Dynamic Bayesian Multinets. In *Proceedings of the 16th conf. on Uncertainty in Artificial Intelligence.* Morgan Kaufmann, 2000.

[16] J.A. Bilmes. Factored sparse inverse covariance matrices. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, 2000.

[17] J.A. Bilmes and K. Kirchhoff. Directed graphical models of classifier combination: Application to phone recognition. In *Proc. Int. Conf. on Spoken Language Processing*, Beijing, China, 2000.

[18] C. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[19] H. Bourlard. Personal communication, 1999.

[20] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, 1994.

[21] P.F. Brown. *The Acoustic Modeling Problem in Automatic Speech Recognition.* PhD thesis, CMU, 1987.

[22] P.F. Brown, V.J. Delle Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[23] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Trans. on Knowledge and Data Engineering*, 8:195–210, 1994.

[24] K.P. Burnham and D.R. Anderson. *Model Selection and Inference : A Practical Information-Theoretic Approach*. Springer-Verlag, 1998.

[25] R. Chellappa and A. Jain, editors. *Markov Random Fields: Theory and Application*. Academic Press, 1993.

[26] Francine R. Chen. Identification of contextual factors for pronunciation networks. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 753–756, 1990.

[27] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco, 1996. Association for Computational Linguistics, Morgan Kaufmann Publishers.

[28] D.M. Chickering. *Learning from Data: Artificial Intelligence and Statistics*, chapter Learning Bayesian networks is NP-complete. Springer Verlag, 1996.

[29] G. Cooper and E. Herskovits. Computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[30] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. McGraw Hill, 1990.

[31] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[32] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[33] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is np-hard. *Artificial Intelligence*, 60(141-153), 1993.

[34] Journal: Data mining and knowledge discovery. Kluwer Academic Publishers. Maritime Institute of Technology, Maryland.

[35] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. *AAAI*, pages 524–528, 1988.

[36] J.R. Deller, J.G. Proakis, and J.H.L. Hansen. *Discrete-time Processing of Speech Signals*. MacMillan, 1993.

[37] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B.*, 39, 1977.

[38] L. Deng, M. Aksmanovic, D. Sun, and J. Wu. Speech recognition using hidden Markov models with polynomial regression functions as non-stationary states. *IEEE Trans. on Speech and Audio Proc.*, 2(4):101–119, 1994.

[39] L. Deng and C. Rathinavelu. A Markov model containing state-conditioned second-order non-stationarity: application to speech recognition. *Computer Speech and Language*, 9(1):63–86, January 1995.

[40] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., 2000.

[41] E. Eide. Automatic modeling of pronunciation variations. In *Eurospeech-99*, 1999.

[42] K. Elenius and M. Blomberg. Effects of emphasizing transitional or stationary parts of the speech signal in a discrete utterance recognition system. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 535–538. IEEE, 1982.

[43] J.G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.

[44] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugen.*, 7:179–188, 1936.

[45] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, New York, NY, 1980.

[46] V. Fontaine, C. Ris, and J.M.Boite. Nonlinear discriminant analysis for improved speech recognition. In *Proceedings of Eurospeech*, pages 2071–2074, 1997.

[47] E. Fosler-Lussier. *Dynamic Pronunciation Models for Automatic Speech Recognition.* PhD thesis, University of California, Berkeley., 1999.

[48] B. Frey. *Graphical Models for Machine Learning and Digital Communication.* MIT Press, 1998.

[49] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[50] N. Friedman and M. Goldszmidt. *Learning in Graphical Models*, chapter Learning Bayesian Networks with Local Structure. Kluwer Academic Publishers, 1998.

[51] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. *14th Conf. on Uncertainty in Artificial Intelligence*, 1998.

[52] K. Fukunaga. *Introduction to Statistical Pattern Recognition, 2nd Ed.* Academic Press, 1990.

[53] S. Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(2):254–272, April 1981.

[54] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(1):52–59, February 1986.

[55] Sadaoki Furui. On the role of spectral transition for speech perception. *Journal of the Acoustical Society of America*, 80(4):1016–1025, October 1986.

[56] M. J. F. Gales and S. Young. An improved approach to the hidden Markov model decomposition of speech and noise. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages I–233–236, 1992.

[57] M.J.F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, May 1999.

[58] M.J.F. Gales and S.J. Young. Segmental hidden Markov models. *Proc. Euro. Conf. Speech Commun. Technol. (EUROSPEECH93)*, pages 1579–1582, 1993.

[59] M.J.F. Gales and S.J. Young. Robust speech recognition in additive and convolutional noise using parallel model combination. *Computer Speech and Language*, 9:289–307, 1995.

[60] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

[61] Z. Ghahramani. *Lecture Notes in Artificial Intelligence*, chapter Learning Dynamic Bayesian Networks. Springer-Verlag, 1998.

[62] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29, 1997.

[63] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins, 1996.

[64] R.M. Gray and A. Gersho. *Vector Quantization and Signal Compression*. Klewer, 1991.

[65] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice*. Prentice Hall, 1993.

[66] X.F. Guo, W.B. Zhu, Q. Shi, S. Chen, and R. Gopinath. The IBM LVCSR system used for 1998 mandarin broadcast news transcription evaluation. In *The 1999 DARPA Broadcast News Workshop*, 1999.

[67] A.K. Halberstadt and J.R. Glass. Heterogeneous measurements and multiple classifiers for speech recognition. *Proceedings ICSLP-98*, pages 995–998, 1998.

[68] D.A. Harville. *Matrix Algebra from a Statistician's Perspective*. Springer, 1997.

[69] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society series B*, 58:158–176, 1996.

[70] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft, 1995.

[71] D. Heckerman, Max Chickering, Chris Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for density estimation, collaborative filtering, and data visualization. In *Proceedings of the 16th conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 2000.

[72] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft, 1994.

[73] H. Hermansky, D. Ellis, and S. Sharma. Tandem connectionist feature stream extraction for conventional hmm systems. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, 2000.

[74] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Allan M. Wylde, 1991.

[75] X.D. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, 2001.

[76] T.S. Jaakkola and M.I. Jordan. *Learning in Graphical Models*, chapter Improving the Mean Field Approximations via the use of Mixture Distributions. Kluwer Academic Publishers, 1998.

[77] R.A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888, 1995.

[78] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[79] F.V. Jensen. *An Introduction to Bayesian Networks*. Springer, 1996.

[80] M.I. Jordan and C. M. Bishop, editors. *An Introduction to Graphical Models*. to be published, 2001.

[81] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. *Learning in Graphical Models*, chapter An Introduction to Variational Methods for Graphical Models. Kluwer Academic Publishers, 1998.

[82] M.I. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[83] B.-H. Juang and L.R. Rabiner. Mixture autoregressive hidden Markov models for speech signals. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 33(6):1404–1413, December 1985.

[84] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[85] M. Kadirkamanathan and A.P. Varga. Simultaneous model re-estimation from contaminated data by composed hidden Markov modeling. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 897–900, 1991.

[86] T. Kamm, G. Andreou, and J. Cohen. Vocal tract normalization in speech recognition compensating for systematic speaker variability. In *Proc. of the 15th Annual speech research symposium*, pages 175–178. CLSP, Johns Hopkins University, 1995.

[87] J. Karhunen. Neural approaches to independent component analysis and source separation. In *Proc 4th European Symposium on Artificial Neural Networks (ESANN '96)*, 1996.

[88] P. Kenny, M. Lennig, and P. Mermelstein. A linear predictive HMM for vector-valued observations with applications to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(2):220–225, February 1990.

[89] B.E.D. Kingsbury and N. Morgan. Recognizing reverberant speech with RASTA-PLP. *Proceedings ICASSP-97*, 1997.

[90] K. Kirchhoff. Combining acoustic and articulatory information for speech recognition in noisy and reverberant environments. In *Proceedings of the International Conference on Spoken Language Processing*, 1998.

[91] K. Kirchhoff and J. Bilmes. Dynamic classifier combination in hybrid speech recognition systems using utterance-level confidence values. *Proceedings ICASSP-99*, pages 693–696, 1999.

[92] J. Kittler, M. Hataf, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[93] K. Kjaerulff. Triangulation of graphs - algorithms giving small total space. Technical Report R90-09, Department of Mathematics and Computer Science. Aalborg University., 1990.

[94] P. Krause. Learning probabilistic networks. *Philips Research Labs Tech. Report.*, 1998.

[95] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems 7*. MIT Press, 1995.

[96] F. R. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):498–519, 2001.

[97] N. Kumar. *Investigation of Silicon Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*. PhD thesis, Johns Hopkins University, 1997.

[98] S.L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

[99] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9:171–185, 1995.

[100] E. Levin. Word recognition using hidden control neural architecture. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 433–436. IEEE, 1990.

[101] E. Levin. Hidden control neural architecture modeling of nonlinear time varying systems and its applications. *IEEE Trans. on Neural Networks*, 4(1):109–116, January 1992.

[102] H. Linhart and W. Zucchini. *Model Selection*. Wiley, 1986.

[103] B.T. Logan and P.J. Moreno. Factorial HMMs for acoustic modeling. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1998.

[104] D.J.C. MacKay. *Learning in Graphical Models*, chapter Introduction to Monte Carlo Methods. Kluwer Academic Publishers, 1998.

[105] J. Makhoul. Linear prediction: A tutorial review. *Proc. IEEE*, 63:561–580, April 1975.

[106] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[107] G.J. McLachlan. *Finite Mixture Models*. Wiley Series in Probability and Statistics, 2000.

[108] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics, 1997.

[109] C. Meek. Causal inference and causal explanation with background knowledge. In Besnard, Philippe and Steve Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 403–410, San Francisco, CA, USA, August 1995. Morgan Kaufmann Publishers.

[110] M. Meila. *Learning with Mixtures of Trees*. PhD thesis, MIT, 1999.

[111] M. Mohri, F. C. N. Pereira, and M. Riley. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32, 2000.

[112] N. Morgan and B. Gold. *Speech and Audio Signal Processing*. John Wiley and Sons, 1999.

[113] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

[114] H.J. Nock and S.J. Young. Loosely-coupled HMMs for ASR. In *Proc. Int. Conf. on Spoken Language Processing*, Beijing, China, 2000.

[115] M. Ostendorf, V. Digalakis, and O. Kimball. From HMM's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Trans. Speech and Audio Proc.*, 4(5), September 1996.

[116] M. Ostendorf, A. Kannan, O. Kimball, and J.. Rohlicek. Continuous word recognition based on the stochastic segment model. *Proc. DARPA Workshop CSR*, 1992.

[117] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd printing edition, 1988.

[118] J. Pearl. *Causality*. Cambridge, 2000.

[119] M.P. Perrone and L.N. Cooper. When networks disagree: ensemble methods for hybrid neural networks. In R.J. Mammone, editor, *Neural Networks for Speech and Image Processing*, page Chapter 10, 1993.

[120] J. Picone, S. Pike, R. Regan, T. Kamm, J. Bridle, L. Deng, Z. Ma, H. Richards, and M. Schuster. Initial evaluation of hidden dynamic models on conversational speech. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1999.

[121] S.D. Pietra, V.D. Pietra, and J. Lafferty. Inducing features of random fields. Technical Report CMU-CS-95-144, CMU, May 1995.

[122] A.B. Poritz. Linear predictive hidden Markov models and the speech signal. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1291–1294, 1982.

[123] A.B. Poritz. Hidden Markov models: A guided tour. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 7–13, 1988.

[124] L.R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, 1993.

[125] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 1986.

[126] M. Richardson, J. Bilmes, and C. Diorio. Hidden-articulator markov models for speech recognition. In *Proc. of the ISCA ITRW ASR2000 Workshop*, Paris, France, 2000. LIMSI-CNRS.

[127] M. Richardson, J. Bilmes, and C. Diorio. Hidden-articulator markov models: Performance improvements and robustness to noise. In *Proc. Int. Conf. on Spoken Language Processing*, Beijing, China, 2000.

[128] T. S. Richardson. *Learning in Graphical Models*, chapter Chain Graphs and Symmetric Associations. Kluwer Academic Publishers, 1998.

[129] Michael D. Riley. A statistical model for generating pronunciation networks. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 737–740, 1991.

[130] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, School of Computer Science, CMU, Pittsburgh, PA, April 1994.

[131] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8), 2000.

[132] R. Rosenfeld, S.F. Chen, and X. Zhu. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computers Speech and Language*, 15(1), 2001.

[133] L.K. Saul, T. Jaakkola, and M.I. Jordan. Mean field theory for sigmoid belief networks. *JAIR*, 4:61–76, 1996.

[134] L.K. Saul and M.I. Jordan. Mixed memory markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, 1999.

[135] R.D. Shachter. Bayes-ball: The rational pastime for determining irrelevance and requisite information in belief networks and influence diagrams. In *Uncertainty in Artificial Intelligence*, 1998.

[136] P. Smyth, D. Heckerman, and M.I. Jordan. Probabilistic independence networks for hidden Markov probability models. Technical Report A.I. Memo No. 1565, C.B.C.L. Memo No. 132, MIT AI Lab and CBCL, 1996.

[137] G. Strang. *Linear Algebra and its applications, 3rd Edition*. Saunders College Publishing, 1988.

[138] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley and Sons, 1985.

[139] H. Tong. *Non-linear Time Series: A Dynamical System Approach*. Oxford Statistical Science Series 6. Oxford University Press, 1990.

[140] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[141] A.P. Varga and R.K. Moore. Hidden Markov model decomposition of speech and noise. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 845–848, Alburquerque, April 1990.

[142] A.P. Varga and R.K. Moore. Simultaneous recognition of concurrent speech signals using hidden makov model decomposition. In *Proceedings Eurospeech*, 1991.

[143] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1990.

[144] T. Verma and J. Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1992.

[145] M. Q. Wang and S. J. Young. Speech recognition using hidden Markov model decomposition and a general background speech model. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages I–253–256, 1992.

[146] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, Submitted.

[147] C.J. Wellekens. Explicit time correlation in hidden Markov models for speech recognition. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 384–386, 1987.

[148] C.J. Wellekens. Personal communication, 2001.

[149] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley and Son Ltd., 1990.

[150] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[151] P.C. Woodland. Optimizing hidden Markov models using discriminative output distributions. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1991.

[152] P.C. Woodland. Hidden Markov models using vector linear prediction and discriminative output distributions. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages I–509–512, 1992.

[153] Su-Lin Wu, Michael L. Shire, Steven Greenberg, and Nelson Morgan. Integrating syllable boundary information into speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, Munich, Germany, April 1997. IEEE.

[154] S. Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, 13(5):45–56, September 1996.

[155] K.H. Yuo and H.C. Wang. Joint estimation of feature transformation parameters and gaussian mixture model for speaker identification. *Speech Communications*, 3(1), 1999.

[156] G. Zweig. *Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, U.C. Berkeley, 1998.

[157] G. Zweig and S. Russell. Speech recognition with dynamic Bayesian networks. *AAAI-98*, 1998.