GRAPHICAL MODELS AND AUTOMATIC SPEECH RECOGNITION

JEFFREY A. BILMES*

Abstract.
Graphical models provide a promising paradigm to study both existing and novel techniques for automatic speech recognition. This paper first provides a brief overview of graphical models and their uses as statistical models. It is then shown that the statistical assumptions behind many pattern recognition techniques commonly used as part of a speech recognition system can be described by a graph – this includes Gaussian 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 for speech recognition and language processing can also be simply described by a graph, including many at the acoustic-, pronunciation-, and language-modeling levels. A number of speech recognition techniques born directly out of the graphical-models paradigm are also surveyed. Additionally, this paper includes a novel graphical analysis regarding why derivative (or delta) features improve hidden Markov model-based speech recognition by improving structural discriminability. It also includes an example where a graph can be used to represent language model smoothing constraints. As will be seen, the space of models describable by a graph is quite large. A thorough exploration of this space should yield techniques that ultimately will supersede the hidden Markov model.

Key words. Graphical Models, Bayesian Networks, Automatic Speech Recognition, Hidden Markov Models, Pattern Recognition, Delta Features, Time-Derivative Features, Structural Discriminability, Language Modeling

1. Introduction. Since its inception, the field of automatic speech recognition (ASR) [129, 39, 21, 164, 83, 89, 117, 80] 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 the hidden Markov model [129], 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 de-

---

*Department of Electrical Engineering, University of Washington, Seattle 98195-2500. This material is based upon work supported in part by the National Science Foundation under Grant No. 0093430.

Chapter from "Mathematical Foundations of Speech and Language Processing", Institute of Mathematical Analysis, 2003.

scribed. 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 [70], auto-regressive models [110], and many information-theoretic coding algorithms [53]. 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 a set of computationally efficient algorithms for probability calculations and decision-making. 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 they often have only minor graphical differences. As we will see in this paper, many of the existing statistical techniques in ASR are 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 [10]. 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)[13], 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 2 provides an overview of GMs showing how they relate to standard statistical procedures. It also surveys a number of GM properties (Section 2.5), such as probabilistic inference and learning. Section 3 casts many of the methods commonly used for automatic speech recognition (ASR) as instances of GMs and their associated algorithms. This includes principle component analysis [44], linear discriminant analysis (and its quadratic and heteroscedastic generalizations) [102], factor analysis, independent component analysis, Gaussian 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.

This section briefly reviews graphical models and their associated algorithms — those well-versed in this methodology may wish to skip directly to Section 3.

Broadly speaking, graphical models offer two primary features to those interested in working with statistical systems. First, 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 that associate with a graph a family of probability distributions. Natural signals (those that 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 [122] down to the fine-grained dependencies existing within the neural code [5]. Second, 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 relative to a naive implementation of probabilistic inference.

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 $X \perp\!\!\!\perp Y | Z$ (notation first introduced in [37]) 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 changes and 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 that obviously do not hold for speech [10], but that nonetheless empirically appear benign, and actually beneficial as argued in Section 3.9). Formal properties of conditional independence are described in [159, 103, 122, 37].

A GM [103, 34, 159, 122, 84] 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), encode 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.

2.1. 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 GM semantics. 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) [122, 84],[1] undirected networks (or Markov random fields) [27], factor graphs [53, 101], chain graphs

---

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

[103, 133] which are combinations of directed and undirected GMs, causal models [123], decomposable models (an important sub-family of models [103]), dependency networks [76], and many others. In general, different graph semantics will correspond to different families of distributions, but overlap can exist (meaning a particular distribution might be describable by two graphs with different semantics).

A Bayesian network (BN) [122, 84, 75] is one type of directed GM where the graph edges are directed and acyclic. In a BN, edges point from parent to child nodes, and such graphs implicitly portray factorizations that are simplifications 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) [38, 66, 56] 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 in the direction of increasing time. DBNs are particularly useful to describe time signals such as speech, and as can be seen from Figure 2 many techniques for ASR fall under this or the BN category.

Several equivalent schemata exist that formally define a BN's conditional independence relationships [103, 122, 84]. 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 are *blocked*. A path is blocked if it has a node v with either: 1) the arrows along the path **do not** converge at v (i.e., serial or diverging at v) $v \in C$; or 2) the arrows along the path **do** converge at v, and neither v nor any descendant of v is in C. Note that C can be the empty set in which case d-separation encodes standard statistical independence.

From d-separation, one may compute a list of conditional independence statements made by 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 [103] (a variable is conditionally independent of its non-descendants given its parents), and the Bayes-ball procedure [143] which is a simple algorithm that one can use to read conditional independence statements from graphs, and which is arguably simpler than d-separation. It is assumed henceforth that the reader is familiar with either d-separation or some equivalent rule.

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 a 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 the factorization of potential functions where each potential function operates only on collections of nodes that form a clique in the graph. A clique is a set of nodes that are pairwise connected [84].

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 might 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 that are concisely representable with BNs but that 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 [103]. These models have important properties relating to efficient probabilistic inference (see below).

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 — it is more accurate to say that there is an edge between A and B than to say that A and B are dependent.

Originally BNs were designed to represent causation, but more recently, models with semantics [123] more precisely representing causality have been developed. Other directed graphical models have been designed as well [76], and can be thought of as the general family of directed graphical models (DGMs).

2.2. 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 (see the left two models in Figure 1). In this case, multiple graphs will correspond to the same family of probability distributions. In such cases, the various GMs are said to be Markov equivalent [153, 154, 77]. In general, it is not immediately obvious with complicated graphs how to visually determine if Markov equivalence holds, but algorithms are available that can determine the members of an equivalence class

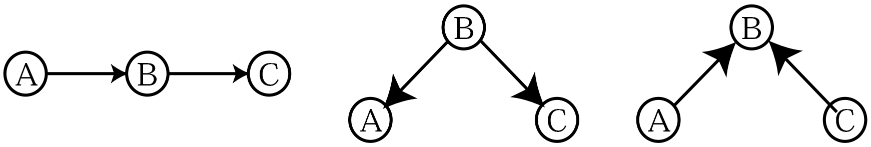


FIGURE 1. *This figure shows four BNs with different arrow directions over the same random variables, A, B, and C. On the left side, the variables form a three-variable first-order Markov chain* $A \rightarrow B \rightarrow C$. *In the middle graph, the same conditional independence statement is realized even though one of the arrow directions has been reversed. Both these networks state that* $A {\perp\!\!\!\perp} C | B$. *The right network corresponds to the property* $A {\perp\!\!\!\perp} C$ *but* **not** *that* $A {\perp\!\!\!\perp} C | B$.

[153, 154, 114, 30].

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 some cases, contains only hidden variables. Note that the graphs in GMs do *not* show the zeros that exist in the stochastic transition matrices of a Markov chain — GMs, rather, encode statistical independence properties of a model (see also Section 3.7).

A node in a graph might 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 [40], 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 not exist since the hidden variables might only be hypothetical (e.g., specified 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. One common way of using

GMs in ASR, in fact, is to use hidden variables to represent some condition known during training and unknown during recognition (see Section 5).

Certain GMs allow for what are called *switching* dependencies [65, 115, 16]. 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. A related construct allows GMs to have optimized local probability implementations [55] using, for example, decision trees.

It is sometimes the case that certain observed variables are used only 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 applications 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 that are marginally independent of a class variable (say C) but that are dependent conditioned on other observations (say X). 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 [75, 77] or frequentist point of view [25, 99, 75]. Structure learning is akin to both statistical model selection [107, 26] and data mining [36]. Several good reviews of structure learning are presented in [25, 99, 75]. Structure learning from a discriminative perspective, thereby producing what is called *discriminative generative models*, was proposed in [10].

Figure 2 depicts a topological hierarchy of both the semantics and structure of GMs, and shows where different models fit in, including several ASR components to be described in Section 3.

2.3. Implementation. When two nodes are connected by a dependency edge, the local conditional probability representation of that dependency may be called its *implementation*. An edge between variable X and Y can represent a lack of independence in a number of ways depending on if the variables are discrete or continuous. For example, one might use discrete conditional probability tables (CPTs) [84], compressed tables [55], decision trees [22], or even a deterministic function (in which case GMs may represent data-flow [1] graphs, or may represent channel coding algorithms [53]). A node in a GM can also depict a constant input parameter

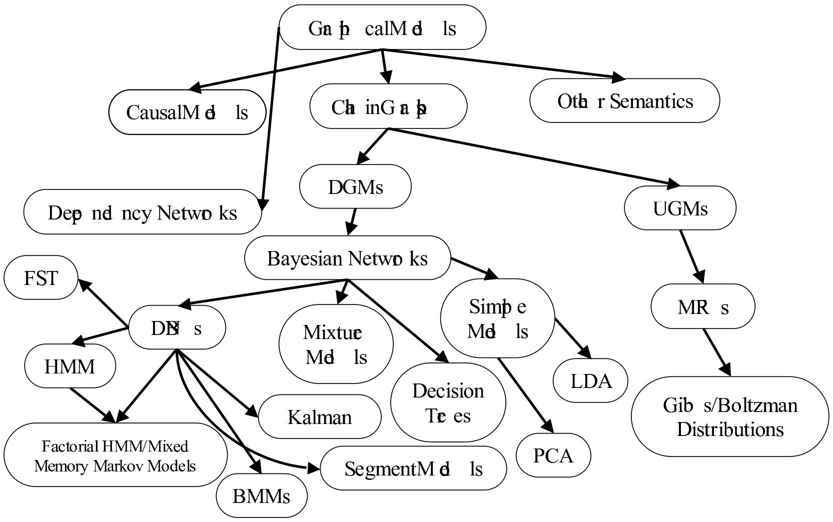

Graphical Models
Causal Models
Chain Graphs
Other Semantics
DGMs
Dependency Networks
UGMs
Bayesian Networks
FST
DBNs
Mixture Models
Simple Models
MRFs
HMM
Decision Trees
LDA
Kalman
Gibbs/Boltzman Distributions
Factorial HMM/Mixed Memory Markov Models
Segment Models
PCA
BMMs

FIGURE 2. *A topology of graphical model semantics and structure*

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 [19], or a STAR [149] or MARS [54] model). In general, different edges in a graph will have different implementations.

In UGMs, conditional distributions are not explicitly represented. Rather a joint distribution over all the variables is constructed using a product of clique potential functions as mentioned in Section 2.1. In general the clique potentials can be arbitrary functions, although certain types are commonly used such as Gibbs or Boltzmann distributions [79]. Many such models fall under what are known as exponential models [44]. The implementation of a dependency in an UGM, therefore, is implicitly specified via these functions in that they specify the way in which subsets of variables, depending on their values, can influence the resulting probability

2.4. 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 random distributions can themselves be seen as nodes, Bayesian approaches are easily represented [75] with GMs.

Many algorithms exist for training the parameters of a graphical model. These include maximum likelihood [44] such as the EM algorithm [40], discriminative or risk minimization approaches [150], gradient descent [19], sampling approaches [109], or general non-linear optimization [50].

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 [16, 47].

2.5. 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 [40, 113]. 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 inference algorithms that can be used.

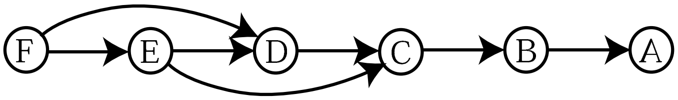


FIGURE 3. *The graph's independence properties are used to move sums inside of factors.*

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 that 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 3, 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 [122, 84]. In fact, the commonly used forward-backward algorithm [129] for hidden Markov models is just a special case of the junction tree algorithm [144], 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 [85], 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 cases 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 (possibly) 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 made by the original DGM. Otherwise, inference might not be correct.

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 [122, 103] adds edges until all cycles in the graph (of length 4 or greater) contain a pair of non-consecutive nodes (along the cycle) that are connected by an edge (i.e., a chord) not part of the cycle edges. 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 [98].

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 [32] 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 — i.e., the algorithm is exponential in the clique sizes, a quantity important to minimize during triangulation. There are many ways to triangulate [98], 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 [84, 122, 34, 85].

Exact inference, such as the above, is useful only for moderately complex networks since inference is NP-hard in general [31]. Approximate inference procedures can, however, be used when exact inference is not feasible. There are several approximation methods including variational techniques [141, 81, 86], Monte Carlo sampling methods [109], and loopy belief propagation [156]. Even approximate inference can be NP-hard however [35]. 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.

3.1. Acoustic Modeling: 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 completely characterize the distribution). In this section, it will be shown how Gaussians can be viewed as both undirected and directed GMs, and how each of these views describe distinct properties of the Gaussian.

An N-dimensional Gaussian density has the form:

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

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 [68], 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 and less well-known 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, $K_{AB} = 0$, a property that 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 [111, 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 that has a fixed variance for all values of x_B but has a mean that is an affine 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 $K_{AA} = \Sigma_{A|B}^{-1}$ [103, 159].

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 $x_{Aa} \perp\!\!\!\perp x_{Ab}$, but 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 $X_i \perp\!\!\!\perp X_j | X_{\{1:N\}\setminus\{i,j\}}$.

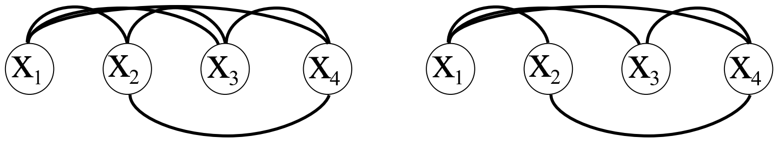


FIGURE 4. *A Gaussian viewed as an UGM. On the left, there are no independence assumptions. On the right,* $X_2 \perp\!\!\!\perp X_3 | \{X_1, X_4\}$.

This property of Gaussians corresponds to their view as an UGM. To see this, first consider a fully connected UGM with N nodes, something that 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 (see [103] for a formal proof where the pairwise Markov property and the global Markov property are related). This is shown in Figure 4. 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:

$$p(x_{1:N}) = \prod_i (2\pi\Sigma_{i|i+1:N})^{-1/2} e^{-\frac{1}{2}(x_i - \mu_{i|i+1:N})^2 \Sigma^{-1}_{i|i+1:N}} \tag{3.1}$$

An identical decomposition of this Gaussian can be produced in a different way. Every concentration matrix K has a unique factorization $K = U^T D U$ where U is a unit upper-triangular matrix and D is diagonal [111, 73]. 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 $K = R^T R$, 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:

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

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

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

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:

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

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

$$p(x_{1:N}) = \prod_i (2\pi)^{-1/2}D_{ii}^{1/2}e^{-\frac{1}{2}(x_i-B_{i,i+1:N}x_{i+1:N}-\tilde{\mu}_i)^2 D_{ii}} \tag{3.2}$$

When this is equated with Equation (3.1), 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 (3.2), 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.[2]

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

---

[2] Standard notation is used here, where if A and B are sets, $A \setminus B$ is the set of elements in A that are not in B.

B matrix, possibly implying different independence statements (depending on if the graphs are Markov equivalent, see Section 2.2). Moreover, a B matrix can be sparse for one ordering, but for a different ordering the B matrix can be dense, and zeros in B might or might not yield zeros in $K = (I - B)^T D(I - B)$ 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 B entries, take $B_{23} = 0$. For this B matrix and when $D = I$, neither the concentration nor the covariance matrix has any zeros, although they are both 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, but in the undirected formalism it is impossible to encode this independence statement and one is forced to generalize and to use a model that possesses no independence properties.

The opposite can occur as well, where zeros exist in K or Σ, and less sparsity exists in the B matrix. Take, for example the matrix,

$$K = \begin{pmatrix} 5 & 2 & 0 & 4 \\ 2 & 9 & 1 & 0 \\ 0 & 1 & 5 & 3 \\ 4 & 0 & 3 & 6 \end{pmatrix}$$

This concentration matrix states that $X_1 \perp\!\!\!\perp X_3 | \{X_2, X_4\}$ and $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$, but the corresponding B matrix has only a single zero in its upper portion reflecting only the first independence statement.

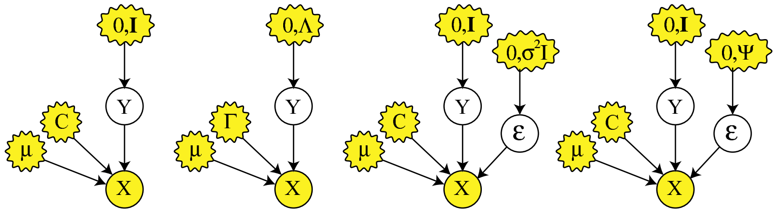
It was mentioned earlier that UGMs and DGMs represent different families of probability distributions, and this is reflected in the Gaussian case above by a reduction in sparsity when moving between certain B and K matrices. It is interesting to note that Gaussians are able to represent any of the dependency 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.

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 a representation. This was exploited in [17], 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.

Lastly, 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. Acoustic Modeling: 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) (also called the Karhunen-Loéve or KL transform), factor analysis (FA), and independent component analysis (ICA). The PCA technique is often presented without any probabilistic interpretation. Interestingly, when given such an interpretation and seen as a graph, PCA has exactly the same structure as both FA and ICA – the only difference lies in the implementation of the dependencies.

The graphs in this and the next section show nodes both for 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. Parameter nodes will be indicated as shaded rippled circles. For our purposes, these nodes constitute constant random variables whose probability score is not counted (they are conditional-only variables, always to the right of the conditioning bar in a probability equation). In a more general Bayesian setting [77, 75, 139], however, these nodes would be true random variables with their own distributions and hyper-parameters.



PCA: X=CY+μ PCA: X=ΓY+μ PPCA: X=CY+μ+ε FA: X=CY+μ+ε

FIGURE 5. *Left two graphs: two views of principle components analysis (PCA); middle: probabilistic PCA; right; factor analysis (FA). In general, the graph corresponds to the equation* $X = CY + \mu + \epsilon$*, where* $Y \sim \mathcal{N}(0, \Lambda)$ *and* $\epsilon \sim \mathcal{N}(0, \Psi)$*.* X *is a random conditional Gaussian with mean* $CY + \mu$ *and variance* $C\Lambda C^T + \Psi$*. With PCA,* $\Psi = 0$ *so that* $\epsilon = 0$ *with probability 1. Also, either (far left)* $\Lambda = I$ *is the identity matrix and* C *is general, or (second from left)* Λ *is diagonal and* $C = \Gamma$ *is orthonormal. With PPCA,* $\Psi = \sigma^2 I$ *is a spherical covariance matrix, with diagonal terms* σ^2*. With FA,* Ψ *is diagonal. Other generalizations are of possible, but they can lead to an indeterminacy of the parameters.*

Starting with PCA, observations of a d-dimensional random vector X

are assumed to be Gaussian with mean μ and covariance Σ. The goal of PCA is to produce a vector Y that is a zero-mean uncorrelated linear transformation of X. The spectral decomposition theorem [146] yields the factorization $\Sigma = \Gamma\Lambda\Gamma^T$, where Γ is an orthonormal rotation matrix (the columns of Γ are orthogonal eigenvectors, each having unit length), and Λ is a diagonal matrix containing the eigenvalues that correspond to the variances of the elements of X. A transformation achieving PCA's goal is $Y = \Gamma^T(X - \mu)$. This follows since $E[YY^T] = \Gamma^T E[(X-\mu)(X-\mu)^T]\Gamma$ $= \Gamma^T\Sigma\Gamma = \Lambda$. Alternatively, a spherically distributed Y may be obtained by the following transformation: $Y = (\Lambda^{-1/2}\Gamma)^T(X-\mu) = C^T(X-\mu)$ with $C = \Lambda^{-1/2}\Gamma$.

Solving for X as a function of Y yields the following:

$$X = \Gamma Y + \mu$$

Slightly abusing notation, one can say that $X \sim \mathcal{N}(\Gamma Y + \mu, 0)$, meaning that X, conditioned on Y, is a linear-conditional constant "Gaussian" — i.e., a conditional-Gaussian random variable with mean $\Gamma Y + \mu$ and zero variance.[3] In this view of PCA, Y consists of the latent or hidden "causes" of the observed vector X, where $Y \sim \mathcal{N}(0, \Lambda)$, or if the C-transformation is used above, $Y \sim \mathcal{N}(0, I)$ where I is the identity matrix. In either case, the variance in X is entirely explained by the variance within Y, as X is simply a linear transformation of these underlying causes. PCA transforms a given X to the most likely values of the hidden causes. This is equal to the conditional mean $E[Y|X] = \Gamma^T(X-\mu)$ since $p(y|x) \sim \mathcal{N}(\Gamma^T(x-\mu), 0)$.

The two left graphs of Figure 5 show the probabilistic interpretations of PCA as a GM, where the dependency implementations are all linear. The left graph corresponds to the case where Y is spherically distributed. The hidden causes Y are called the "principle components" of X. It is often the case that only the components (i.e., elements of Y) corresponding to the largest eigenvalues of Σ are used in the model, the other elements of Y are removed, so that Y is k-dimensional with $k < d$. There are many properties of PCA [111] – for example, using the principle k elements of Y leads to the smallest reconstruction error of X in a mean-squared sense. Another notable property (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 components Y. In this sense, PCA explains the variance in X using only variances found in the hidden causes Y.

Factor analysis (the right-most graph in Figure 5) is only a simple modification of PCA — a single random variable is added onto the PCA equation above, yielding:

$$X = CY + \mu + \epsilon$$

---

[3] This of course corresponds to a degenerate Gaussian, as the covariance matrix is singular.

where $Y \sim \mathcal{N}(0, I)$, and $\epsilon \sim \mathcal{N}(0, \Psi)$ with Ψ a non-negative diagonal matrix. In factor analysis, C is the *factor loading* matrix and Y the *common factor* vector. Elements of the residual term $\epsilon = X - CY - \mu$, are called the *specific factors*, and account both for noise in the model and for the underlying variance in X. In other words, X possesses a non-zero variance, even conditional on Y, and Y is constrained to be unable to explain the variance in X since Y is forced to have I as a covariance matrix. C, on the other hand, is compelled to represent just the correlation between elements of X irrespective of its individual variance terms, since correlation can not be represented by ϵ. Therefore, unlike PCA, if the scale of an element of X changes, the resulting Y will not change as it is ϵ that will absorb the change in X's variance. As in PCA, it can be seen that in FA X is being explained by underlying hidden causes Y, and the same graph (Figure 5) can describe both PCA and FA.

Probabilistic PCA (PPCA) (second from the right in Figure 5) [147, 140] while not widely used in ASR is only a simple modification to FA, where $\Psi = \sigma^2 I$ is constrained so that ϵ is a spherically-distributed Gaussian.

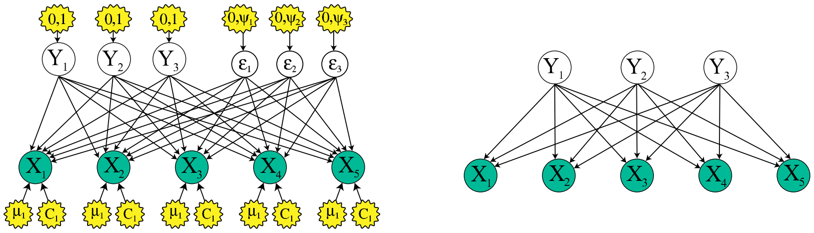


FIGURE 6. *Left: A graph showing the explicit scalar variables (and therefore their statistical dependencies) for PCA, PPCA, and FA. The graph also shows the parameters for these models. In this case, the dependencies are linear and the random variables are all Gaussian. Right: The graph for PCA/PPCA/FA (parameters not shown) which is the same as the graph for ICA. For ICA, the implementation of the dependencies and the random variable distributions can be arbitrary, different implementations lead to different ICA algorithms. The key goal in all cases is to explain the observed vector X with a set of statistically independent causes Y.*

In all of the models above, the hidden causes Y are uncorrelated Gaussians, and therefore are marginally independent. Any statistical dependence between elements of X exist only in how they are jointly dependent on one or more of the hidden causes Y. It is possible to use a GM to make this marginal Y dependence explicit, as is provided on the left in Figure 6 where all nodes are now scalars. In this case, $Y_j \sim \mathcal{N}(0, 1)$, $\epsilon_j \sim \mathcal{N}(0, \psi_j)$, and $p(x_i) \sim \mathcal{N}(\sum_j C_{ij} y_j + \mu_i, \psi_i)$ where $\psi_j = 0$ for PCA.

The PCA/PPCA/FA models can be viewed without the parameter and noise nodes, as shown on the right in Figure 6. This, however, is the general model for independent component analysis (ICA) [7, 92], another method that explains data vectors X with independent hidden causes. Like

PCA and FA, a goal of ICA is to first learn the parameters of the model that explain X. Once done, it is possible to find Y, the causes of X, that are as statistically independent as possible. Unlike PCA and FA, however, dependency implementations in ICA neither need to be linear nor Gaussian. Since the graph on the right in Figure 6 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 $E[Y|X]$. Inference can still be done using the standard graphical-model inference machinery, described in Section 2.5.

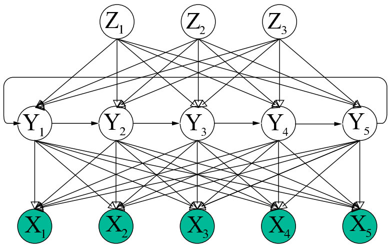


FIGURE 7. *Multi-level ICA*

.

Further generalizations of PCA/FA/ICA can be obtained simply by using different implementations of the basic graph given in Figure 6. 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 7, can easily be described where the middle hidden layer is a possibly non-independent explanation for the final marginally independent components. The goal again is to train parameters to explain X, and to compute $E[Z|X]$. With graphs it is therefore easy to understand all of these techniques, and simple structural or implementation changes can lead to dramatically different statistical procedures.

3.3. Acoustic Modeling: LDA/QDA/MDA/QMDA. When the goal is pattern classification [44] (deciding amongst a set of classes for X), it is often beneficial to first transform X to a space spanned neither by the principle nor the independent components, but rather to a space that best discriminatively represents the classes. Let C be a variable that indicates the class of X, $|C|$ the cardinality of C. As above, a linear transformation can be used, but in this case it is created to maximize the between-class covariance while minimizing the within-class covariance in the transformed space. Specifically, the goal is to find the linear transfor-

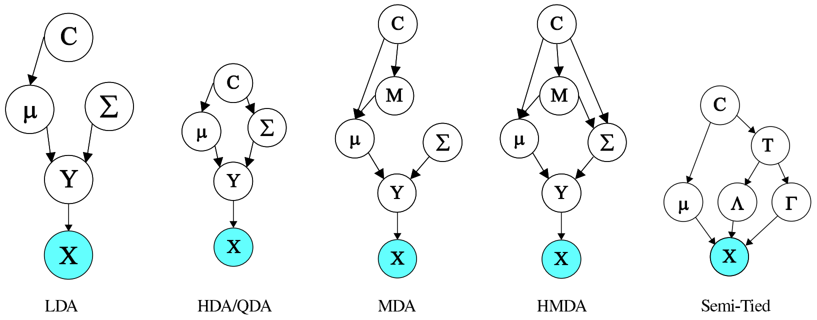


FIGURE 8. *Linear discriminant analysis (left), and its generalizations.*

mation matrix A to form $Y = AX$ that maximizes $\mathrm{tr}(BW^{-1})$ [57] where

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

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) [49].

LDA can also be seen as a particular statistical modeling assumption about the way in which observation samples X are generated. In this case, it is assumed that the class conditional distributions in the transformed space $P(Y|C = i)$ are Gaussians having priors $P(C = i)$. Therefore, 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:

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

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 their different means. Finally, it is assumed that there is a linear transform relating X to Y. The goal of LDA is to find the linear transformation that maximizes the likelihood $P(X)$ under the assumptions given by the model above. The statistical model behind LDA can therefore be graphically described as shown on the far left in Figure 8.

There is a intuitive way in which these two views of LDA (a statistical model or simply an optimizing linear transform) can be seen as identical. Consider two class-conditional Gaussian distributions with identical covariance matrices. In this case, the discriminant functions are linear, and effectively project any unknown sample down to an affine set[4], in this case a line, that points in the direction of the difference between the two means [44]. It is possible to discriminate as well as possible by choosing a threshold along this line — the class of X is determined by the side of the threshold X's projection lies.

More generally, consider the affine set spanned by the means of $|C|$ class-conditional Gaussians with identical covariance matrices. Assuming the means are distinct, this affine set has dimensionality $\min\{|C|-1, \dim(X)\}$. Discriminability is captured entirely within this set since the decision regions are hyperplanes orthogonal to the lines containing pairs of means [44]. The linear projection of X onto the $|C|-1$ dimensional affine set Y spanned by the means leads to no loss in classification accuracy, assuming Y indeed is perfectly described with such a mixture. If fewer than $C-1$ dimensions are used for the projected space (as is often the case with LDA), this can lead to a dimensionality reduction algorithm that has a minimum loss in discriminative information. It is shown in [102] that the original formulation of LDA ($Y = AX$ above) is identical to the maximum likelihood linear transformation from the observations X to Y under the model described by the graph shown on the left in Figure 8.

When LDA is viewed as graphical model, it is easy to extend it 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 GM second from the left in Figure 8. This has been called quadratic discriminant analysis (QDA) [44, 113], because decision boundaries are quadratic rather than linear, or heteroscedastic discriminant analysis (HDA) [102], 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 remainder corresponds in the projected space to the dimensions that do not carry discriminative information.

Further generalizations to LDA are immediate. For example, if the class conditional distributions are Gaussians mixtures, every component sharing the same covariance matrix, then mixture discriminant analysis (MDA) [74] is obtained (3rd from the left in Figure 8). A further generalization yields what could be called be called heteroscedastic MDA, as described 2nd from the right in Figure 8. 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 [51, 95, 78] that have recently been used.

Taking note of the various factorizations one may perform on a positive-definite matrix [73], a concentration matrix K within a Gaussian distribution can be factored as $K = \Lambda^T \Gamma \Lambda$. 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 [62, 165]. Once again, this form of tying can be described by a GM as shown by the far right graph in Figure 8.

---

[4] An affine set is simply a translated subspace [135].

3.4. Acoustic Modeling: 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 that determines the active mixture component 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 $C \to X$. Conditional mixture generalizations, where X requires Z, are quite easy to obtain using the graph $Z \to C \to X$, 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 such as [148, 112] describe the properties of mixture distributions, most of which can be described using graphs in this way.

3.5. Acoustic Modeling: 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 [82, 100, 124, 160, 19]. 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 [12, 94, 72, 95], the HMM state level [96], the sub-word or word level [163], and even at the utterance level [48].

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 [97] where $p(c|x) = \sum_i p(i)p_i(c|x)$, 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 that leads to the particular combination rule. Ideally, the combination rule that performs best will correspond to the model that 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 independence assumptions corresponding to a naive Bayes classifier [18].

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.

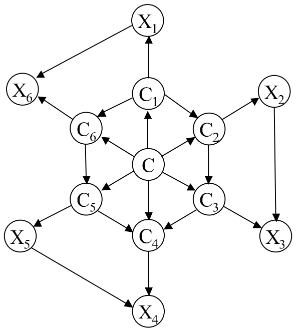


FIGURE 9. *A GM to describe the process of classifier combination. The model does not require in all cases that the errors are statistically independent.*

More advanced combination schemes can be defined by, of course, assuming more sophisticated models. One such example is shown in Figure 9, 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 match the data available and that need not make assumptions which might not be true.

3.6. Acoustic Modeling: 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) [91], and explicit parameter adaptation such as maximum-likelihood linear regression (MLLR) [104]. It turns out that these procedures may also be described with a GM.

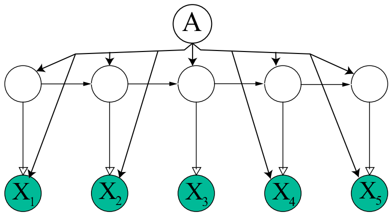


FIGURE 10. *A GM to describe various adaptation and global parameter transformation methods, such as VTLN, MLLR, and SAT. The variable A indicates that the parameters of the entire model can be adapted.*

VTLN corresponds to augmenting an HMM model with an additional global hidden variable that indicates the vocal tract length. This variable determines the transformation on the acoustic feature vectors that should be performed to “normalize” the affect of vocal-tract length on these features. It is common in VTLN to perform all such transformations, and the one yielding the highest likelihood of the data is ultimately chosen to produce a probability score. The graph in Figure 10 shows this model, where A indicates vocal-tract length and that can potentially affect the entire model as shown (in this case, the figure shows a hidden Markov model which will be described in Section 3.9). In a “Viterbi” approach, only the most probable assignment of A is used to form a probability score. Also, A is often a conditional-only variable (see Section 2.2) so the prior $P(A)$ is not counted. If a prior is available, it is also possible to integrate over all values to produce the final probability score.

MLLR [104], or more generally speaker adaptation [164], corresponds to adjusting parameters of a model at test time using adaptation data that is not available at training time. In an ASR system, this takes the form of training on a speaker or an acoustic environment that is not (necessarily) encountered in training data. Since supervised training requires supervisory information, either that is available (supervised speaker adaptation), or an initial recognition pass is performed to acquire hypothesized answers for an unknown utterance (unsupervised speaker adaptation) — in either case, these hypotheses are used as the supervisory information to adjust the model. After this is done, a second recognition pass is performed. The entire procedure may also be repeated. The amount of novel adaptation information is often limited (typically a single utterance), so rather than adjust all the parameters of the model directly, typically a simple global transformation of those parameters is learned (e.g., a linear transformation of all of the means in a Gaussian-mixture HMM system). This procedure is also described in Figure 10, where A in this case indicates the global transformation. During adaptation, all of the model parameters are held fixed except for A which is adjusted to maximize the likelihood of the adaptation data.

Finally, speaker adaptive training (SAT) [3] is the dual of speaker adaptation. Rather than learn a transformation that maps the parameters from being speaker-independent to being speaker-dependent and doing so at recognition time, in SAT such a transformation is learned at training time. With SAT, the speaker-independent parameters of a model along with speaker-specific transformations are learned simultaneously. This procedure corresponds to a model that possesses a variable that identifies the speaker, is observed during training, and is hidden during testing. The speaker variable is the parent of the transformation mapping from speaker-independent to speaker-dependent space, and the transformation could potentially affect all the remaining parameters in the system. Figure 10 once again describes the basic structure, with A the speaker variable. During recognition, either the most likely transformation can be used (a Viterbi approach), or all speaker transformations can be used to form an integrative score.

In the cases above, novel forms of VTLN, MLLR, or SAT would arise simply by using different implementations of the edges between A and the rest of the model.

3.7. Pronunciation Modeling. 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 [28, 134, 45, 52, 89]. Often these strings are specified probabilistically, where the probability of a given phone depends on the preceding phone (as in a Markov chain), thus producing probabilities of pronunciation variants of a word. The pronunciation may also depend on the acoustics [52].

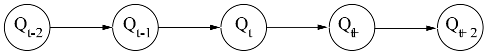


Figure 11. *A simple first-order Markov chain. This graph encodes the relationship* $Q_t \perp\!\!\!\perp Q_{1:t-2} | Q_{t-1}$.

Using the chain rule, the probability of a string of T phones $V_{1:T}$, 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 phones 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 11.

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 represents not only the different phones, but also different positions of different words. Different values of v_t, corresponding to the same phone in different words, would then correspond to the same acoustic Gaussians, but the distribution of v_{t+1} given v_t would be appropriate for the word containing v_t and the position within that word. This procedure is equivalent to turning a K^{th}-order Markov chain into a first-order chain [83].

Another way to achieve this effect is rather than condition on the previous phone, condition instead on the word W_t and the sequential position of a phone in the word S_t, as in $P(V_t|W_t, S_t)$. The position variable is needed to select the current phone. A Markov chain can be used over the two variables W_t and S_t. This approach corresponds to expanding the graph in Figure 11 to one that explicitly mentions the variables needed to keep track of and use each phone, as further described in Section 5.

A GM view of a pronunciation model does not explicitly mention the non-zero entries in the stochastic matrices in the Markov chain. Stochastic finite state automata (SFSA) [130] diagrams are ideally 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 entirely different properties of a Markov chain than do the graphs that are studied in this text.

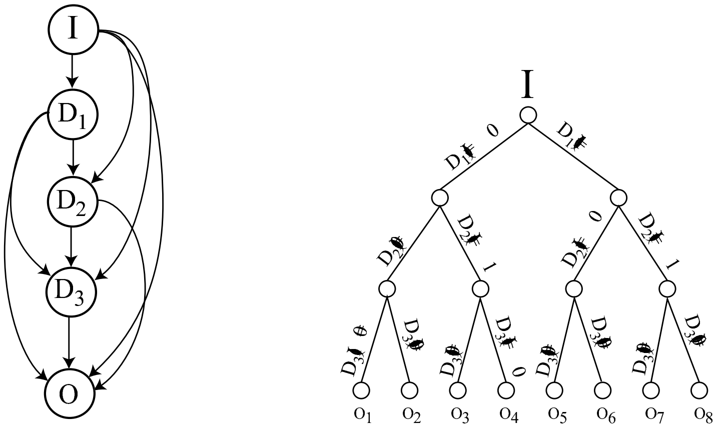


FIGURE 12. *Left: A GM view of a decision tree, which is a probabilistic generalization of the more familiar decision tree on the right.*

Pronunciation modeling often involves a mapping from base-forms (isolated word dictionary-based pronunciations) to surface forms (context-dependent and data-derived pronunciations more likely to correspond to what someone might say). Decision trees are often used for this purpose [28, 134], so it is elucidative at this point to see how they may be described using GMs [86][5]. Figure 12 shows a standard decision tree on the right, and a stochastic GM version of the same decision tree on the left. In the graphical view, there is an input node I, an output node O, and a series of decision random variables D_i. The cardinality of the decision variables D_i is equal to the arity of the corresponding decision tree node (at roughly the same horizontal level in the figure) — the figure shows that all nodes have an arity of two (i.e., correspond to binary random variables).

---

[5] These GMs also describe hierarchical mixtures of experts [87].

In the GM view, the answer to a question at each level of the tree is made with a certain probability. All possible questions are considered, and a series of answers from the top to the bottom of the tree provides the probability of one of the possible outputs of the decision tree. The probability of an answer at a node is conditioned on the set of answers higher in the tree that lead to that node. For example, $D_1 = 0$ means the answer is 0 to the first question asked by the tree. This answer occurs with probability $P(D_1 = i|I)$. The next answer is provided with probability $P(D_2 = j|D_1, I)$ based on the first decision, leading to the graph on the left of the figure.

In a normal decision tree, only one decision is made at each level in the tree. A GM can represent 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 $d_{1:\ell-1}$, where $d_\ell = f_\ell(I, d_{1:\ell-1})$. Therefore, with the appropriate implementation of dependencies, it can be seen that the GM-view is a probabilistic generalization of normal decision trees.

3.8. Language Modeling. Similar to pronunciation modeling, the goal of language modeling is to provide a probability for any possible string of words $W_{1:T}$ in a language. There are many varieties of language models (LMs) [83, 136, 118, 29], and it is beyond the scope of this paper to describe them all. Nevertheless, the following section uses GMs to portray some of the more commonly and successfully used LMs.

At this time, the most common and successful language model is the n-gram. Similar to pronunciation modeling, the chain rule is applied to a joint distribution over words $p(W_{1:T})$. Within each conditional factor $p(W_t|W_{1:t-1})$, the most distant parent variables are dropped until an $(n-1)^{th}$ order Markov chain results $p(W_t|W_{t-n+1:t-1}) = p(W_t|H_t)$, where H_t is the length $n-1$ word history. For a bi-gram ($n = 2$), this leads to a graph identical to the one shown in Figure 11. In general, tri-grams (i.e., 2nd-order Markov chains) have so far been most successful for language modeling among all values n [29].

While a graphical model showing an $(n-1)^{th}$-order Markov chain accurately depicts the statistical independence assumptions made by an n-gram, it does not portray how the parameters of such a model are typically obtained, a procedure that can be quite involved [29]. In fact, much research regarding n-grams involves methods to cope with data-sparsity — because of insufficient training data, "smoothing" methodology must be employed, whereby a K^{th} order model is forced to provide probability for length $K+1$ strings of words that did not occur in training data. If a purely maximum-likelihood procedure was used, these strings would be given zero probability.

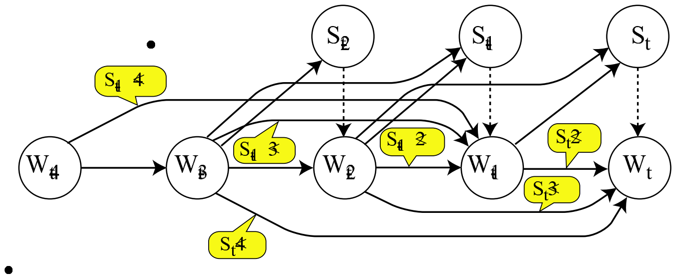


FIGURE 13. *A GM view of a LM. The dashed arcs indicate that the parents are switching. The hidden switching parents* S_t *switch between the word variables* W_t *forming either a zeroth* ($S_t = 1$)*, first* ($S_t = 2$)*, or second* ($S_t = 3$) *order Markov chain. The switching parents also possess previous words as parents so that the probability of the Markov-chain order is itself context dependent.*

Often, smoothing takes the form of mixing together higher- and lower-order sub-models, with mixing weights determined from data not used for training any of the sub-models [83, 29]. In such a case, a language model mixture can be described by the following equation:

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

where $\sum_i \alpha_i = 1$ for all word histories, and where the α coefficients are some (possibly) history-dependent mixing values that determine how much each sub-model should contribute to the total probability score. Figure 13 shows this mixture using a graph with switching parents (see Section 2.2). The variables S_t correspond to the α coefficients, and the edges annotated with values for S_t 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 [83], 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 [29], and such methods are not depicted by Figure 13 (but see Figure 15).

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 [24]. There are a number of ways that these clusters can be formed, such as by grammatical category or by data-driven approaches that

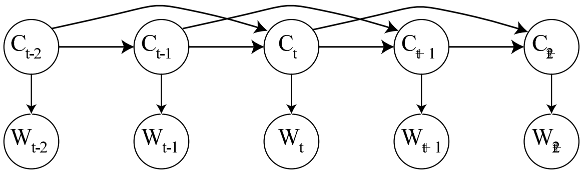


FIGURE 14. *A class-based language model. Here, a Markov chain is used to model the dynamics of word classes rather than the words themselves.*

might use decision trees (as discussed in [83, 24]). Whatever the method, the underlying statistical model can also be described by a GM, as shown in Figure 14. In this figure, a Markov chain exists over a (presumably) much lower dimensional class variables C rather than the high-dimensional word variables. This representation can therefore considerably decrease model complexity and therefore lower parameter estimation variance.

The class-based language model can be further extended to impose certain desirable constraints with respect to words that do not occur in training material, or the so-called unknown words. It is common in a language model to have a special token called `unk` indicating the unknown word. Whenever a word is encountered in a test set that has not occurred in the training set, the probability of `unk` should be given as the probability of the unknown word. The problem, however, is that if maximum likelihood estimates of the parameters of the language model are obtained using the training set, the probability of `unk` will be zero. It is therefore typical to force this token to have a certain non-zero probability and in doing so, essentially "steal" probability mass away from some of the tokens that do indeed occur in the training set. There are many ways of implementing such a feature, generally called language model back-off [83]. For our purposes here, it will be sufficient to provide a simple model, and show how it can be enforced by an explicit graph structure.[6]

Suppose that the vocabulary of words $\mathcal{W}$ can be divided into three disjoint sets: $\mathcal{W} = \{\texttt{unk}\} \bigcup \mathcal{S} \bigcup \mathcal{M}$, where `unk` is the token representing the unknown word, $\mathcal{S}$ is the set of items that have occurred only one time in the training set (the singletons), and $\mathcal{M}$ is the set of all other lexical items. Let us suppose also that we have a maximum-likelihood distribution p_{ml} over words in $\mathcal{S}$ and $\mathcal{M}$, such that $\sum_w p_{ml}(w) = 1$, $p_{ml}(\texttt{unk}) = 0$, and in general

$$p_{ml}(w) = \frac{N(w)}{N},$$

where $N(w)$ is the number of times word w occurs in the training set, and N is the total number of words in the training set. This means, for example, that $N(w) = 1, w \in \mathcal{S}$.

---

[6] Thanks to John Henderson who first posed the problem to me of how to represent this construct using a graphical model, in the context of building a word-tagger.

One possible assumption is to force the probability of `unk` to be 0.5 times the probability of the entire singleton set, i.e., $p(\text{unk}) = 0.5 * p_{ml}(\mathcal{S}) = 0.5 * \sum_{w \in \mathcal{S}} p_{ml}(w)$. This requires that probability be taken away from tokens that do occur in the training set. In this case probability is removed from the singleton words, leading to the following desired probability model $p_d(w)$:

$$p_d(w) = \begin{cases} 0.5 p_{ml}(\mathcal{S}) & \text{if } w = \texttt{unk} \\ 0.5 p_{ml}(w) & \text{if } w \in \mathcal{S} \\ p_{ml}(w) & \text{otherwise} \end{cases} \tag{3.3}$$

This model, of course, can easily be modified so that it is conditioned on the current class $p_d(w|c)$ and so that it uses the conditional maximum likelihood distribution $p_{ml}(w|c)$. Note that this is still a valid probability model, as $\sum_w p_d(w|c) = 1$.

The question next becomes how to represent the constraints imposed by this model using a directed graph. One can of course produce a large conditional probability table that stores all values as appropriate, but the goal here is to produce a graph that explicitly represents the constraints above, and that can be used to train such a model. It might seem at first not to be possible because the variable W must be used both to switch in different distributions and, once a given distribution has been selected, as a probabilistic query variable. The variable W can not exist both on the left of the conditioning bar (where it is possible to produce a probability of $W = w$) and also on the right of the conditioning bar (where it can be used to select the current distribution).

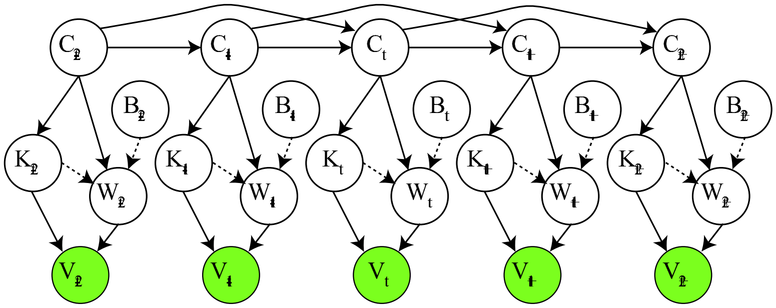


FIGURE 15. *A class-based language model that forces the probability of* `unk` *to be* α *times the probability of all singleton words. The* V_t *variables are shaded, indicating that they are observed, and have value* $V_t = 1, \forall t$. *The* K_t *and* B_t *variables are switching parents of* W_t.

Surprisingly, there exists a solution within the space of directed graphical models shown in Figure 15. This graph, a modified class-based language model, includes a child variable V_t at each time that is always observed to be $V_t = 1$. This means, rather than compute $p(w_t|c_t)$ at each time step, we compute $p(w_t, V_t = 1|c_t)$. The goal is to show that it is possible for $p(w_t, V_t = 1|c_t) = p_d(w_t|c_t)$. The variables B_t and K_t are both binary valued for all t, and these variables are also switching parents (see Section 2.2) of W_t. From the graph, we see that there are a number of conditional distributions that need to be defined. Before doing that, two auxiliary distributions are defined so as to make the definitions that follow simpler:

$$p_M(w|c) \triangleq \begin{cases} \frac{p_{ml}(w|c)}{p(\mathcal{M}|c)} & \text{if } w \in \mathcal{M} \\ 0 & \text{else} \end{cases} \tag{3.4}$$

where $p(\mathcal{M}|c) \triangleq \sum_{w \in \mathcal{M}} p_{ml}(w|c)$, and

$$p_S(w|c) \triangleq \begin{cases} \frac{p_{ml}(w|c)}{p(\mathcal{S}|c)} & \text{if } w \in \mathcal{S} \\ 0 & \text{else} \end{cases} \tag{3.5}$$

where $p(\mathcal{S}|c) \triangleq \sum_{w \in \mathcal{S}} p_{ml}(w|c)$. Note that both p_M and p_S are valid normalized distributions over all words. Also, $p(\mathcal{S}|c) + p(\mathcal{M}|c) = 1$ since these two quantities together use up all the probability mass contained in the maximum likelihood distribution.

The remaining distributions are as follows. First, B_t has a binary uniform distribution:

$$p(B_t = 0) = p(B_t = 1) = 0.5 \tag{3.6}$$

The observation variable $V_t = 1$ simply acts as in indicator, and has a distribution that produces probability one only if certain conditions are met, and otherwise produces probability zero:

$$p(V_t = 1|w_t, k_t) = 1_{\{(w_t \in \mathcal{S}, k_t = 1) \text{ or } (w_t \in \mathcal{M}, k_t = 0) \text{ or } (w_t = \textbf{unk}, k_t = 1)\}} \tag{3.7}$$

where 1_A is a binary indicator function that is unity only when the event A is true, and is zero otherwise. Next, the word distribution will switch between one of three distributions depending on the values of the switching parents K_t and B_t, as follows:

$$p(w_t|k_t, b_t, c_t) = \begin{cases} p_M(w_t|c_t) & \text{if } k_t = 0 \\ p_S(w_t|c_t) & \text{if } k_t = 1 \text{ and } b_t = 1 \\ \delta_{\{w_t = \textbf{unk}\}} & \text{if } k_t = 1 \text{ and } b_t = 0 \end{cases} \tag{3.8}$$

Note that the third distribution is simply a Dirac-delta distribution, giving probability one only when w_t is the unknown word. Last, the distribution for K_t is as follows:

$$p(k_t|c_t) = \begin{cases} p(\mathcal{S}|c_t) & \text{if } k_t = 1 \\ 1 - p(\mathcal{S}|c_t) & \text{otherwise} \end{cases} \tag{3.9}$$

This model correctly produces the probabilities that are given in Equation 3.3. First, when $w_t = \texttt{unk}$:

$$\begin{aligned}
p(w_t = \texttt{unk}, v_t = 1) &= p(v_t = 1 | k_t = 1, w_t = \texttt{unk}) \times p(w_t = \texttt{unk} | k_t = 1, b_t = 0, c_t) \\
&\quad \times p(b_t = 0) \times p(k_t = 1 | c_t) \\
&= 1 \times 1 \times 0.5 \times p(\mathcal{S} | c_t) \\
&= 0.5 p(\mathcal{S} | c_t)
\end{aligned}$$

as desired. This follows because the other terms, for different values of the hidden variables, are all zero. Next, when $w_t \in \mathcal{S}$,

$$\begin{aligned}
p(w_t, v_t = 1) &= p(v_t = 1 | k_t = 1, w_t \in \mathcal{S}) \times p(w_t | k_t = 1, b_t = 1, c_t) \\
&\quad \times p(b_t = 1) \times p(k_t = 1 | c_t) \\
&= 1 \times p_S(w_t | c_t) \times 0.5 \times p(\mathcal{S} | c_t) \\
&= 0.5 * p_{ml}(w_t | c_t)
\end{aligned}$$

again as desired. Lastly, when $w_t \in \mathcal{M}$,

$$\begin{aligned}
p(w_t, v_t = 1) &= p(v_t = 1 | k_t = 0, w_t \in \mathcal{M}) \times \left(\sum_{b_t \in \{0,1\}} p(w_t | k_t = 0, b_t, c_t) p(b_t) \right) \\
&\quad \times p(k_t = 0 | c_t) \\
&= 1 \times p_M(w_t | c_t) \times p(\mathcal{M} | c_t) \\
&= p_{ml}(w_t | c_t).
\end{aligned}$$

In this last case, B_t has no influence as it is marginalized away — this is because the event $K_t = 0$ removes the parent B_t from W_t. Once the graph structures and implementations are set up, standard GM learning algorithms can be used to obtain smoothed parameters for this distribution.

Many other such models can be described by directed graphs in a similar way. Moreover, many language models are members of the family of exponential models[44]. These include those models whose parameters are learned by maximum entropy methods [126, 83, 9, 137], 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 that otherwise has maximum entropy (or minimum KL-divergence with some desired distribution [126]). Note that such an approach can also be used to describe the distribution over an entire sentence[138] at a time, rather than a conditional distribution of the current word w_t given the current history h_t. 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 interesting.

Overall, however, 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.

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 that can be "flattened" down to a single first-order Markov chain [116, 83]. This Markov chain consists of a sequence of serially connected discrete hidden variables during recognition, thus the name HMM.

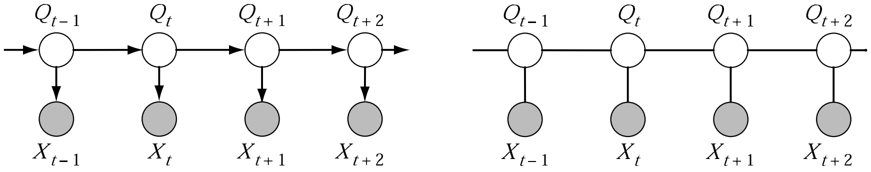


FIGURE 16. *A hidden Markov model (HMM), viewed as a graphical model. Note that an HMM may be equivalently viewed either as a directed (left) or an undirected (right) model, as in this case the conditional independence properties are the same.*

Most generally, a hidden Markov model (HMM) is collection of T discrete scalar random variables $Q_{1:T}$ and T other variables $X_{1:T}$ that may be either discrete or continuous (and either scalar- or vector-valued). These variables, collectively, possess the following conditional independence properties:

$$Q_{t:T} \perp\!\!\!\perp Q_{1:t-2} | Q_{t-1} \tag{3.10}$$

and

$$X_t \perp\!\!\!\perp \{Q_{\neg t}, X_{\neg t}\} | Q_t \tag{3.11}$$

for each $t \in 1:T$. $Q_{\neg t}$ refers to all variables Q_τ except for the one at time $\tau = t$. 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 GM shown in Figure 16. 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 [70] where all 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).

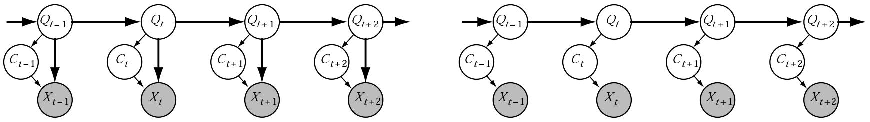


FIGURE 17. *An HMM with mixture observation distributions (left) and a semi-continuous HMM (right).*

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 17). 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 [69]), 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 a mixture with components from this globally shared set of distributions. The GM for such an HMM loses an edge between Q and X as shown on the right in Figure 17. In this case, all of the represented dependence occurs via the hidden mixture variable at each time.

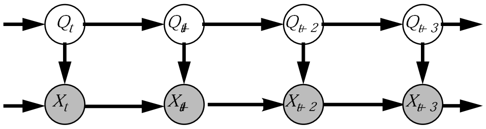


FIGURE 18. *An Auto-regressive HMM as a GM*

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 [157, 23, 120], 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 $K = 1$ is shown in Figure 18. When the additional dependencies are linear and Gaussian, these are sometimes called conditional Gaussian HMMs [120].

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 [127, 128, 88, 129]. These latter models are HMMs that have been inspired from the use of linear-predictive coefficients for speech [129]. 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 that depicts state-conditioned auto-regressive models at the speech sample level.

Our last example of an augmented HMM is something often called an input-output HMM [8] (See Figure 20). 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.

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. Section 4 begins to explore more advanced speech models as viewed from a GM perspective.

3.10. Why Delta Features Work. 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 [46, 58, 59, 60], or just "dynamic" features). Most often, estimates of the derivative are obtained using linear regression [129], 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 as 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 that consists of x_t and $\dot{x}_t$ appended together.

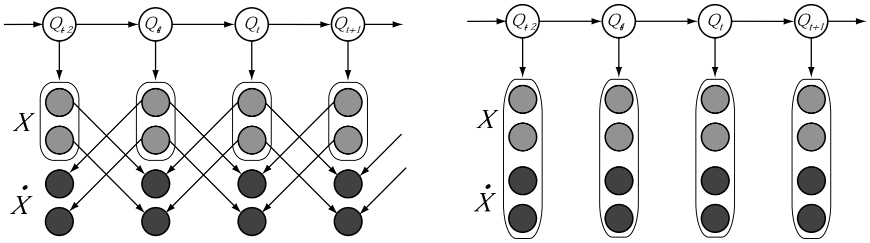


FIGURE 19. *An GM-based explanation of why delta features work in HMM-based ASR systems. The left figure gives a GM that shows the generative process of HMMs with delta features. The right figure shows how delta features are typically used in an HMM system, where the information between $\dot{X}_t$ and 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 [129, 11], 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 19, 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 18), 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 19 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 $Q_{1:T}$. The generative model on the left in Figure 19 shows, however, that there is zero information between the $\dot{X}_t$ and Q_t. When the edges between $\dot{X}_t$ and its parents parents($\dot{X}_t$) are removed, the mutual information [33] between $\dot{X}_t$ and Q_t can only *increase* (from zero to something greater) relative to the generative model. The right of Figure 19 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 has a structure that 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.

Therefore, the very conditional independence assumptions that are commonly seen as a flaw of the HMM provide a benefit when using delta features. More strongly put, the *incorrect* statistical independence properties made by the HMM model on the right of Figure 19 (relative to truth, as shown by the generative model on the left) are the very thing that enable delta features to decrease recognition error. The standard HMM model with delta features seem to be an instance of a model with an inherently discriminative structure [16, 47] (see also Section 5).

In general, can the removal of edges or additional processing lead to and overall increase in the information between the entire random vectors $X_{1:T}$ and $Q_{1:T}$? The data processing inequality [33] says it can not. In the above, each feature vector $(\dot{X}_t, X_t)$ will have more information about the temporally local hidden variable Q_t — this 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 PCA or LDA preprocessing over multiple windows [71] and other non-linear generalizations [51, 95, 78].

It has often been found that conditionally Gaussian HMMs (as in Figure 18) often do not provide an improvement when delta features are included in the feature stream [20, 23, 93, 158]. 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 [161, 162] where discriminative output distributions were used. In [105, 106], 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 [16] (to be described below) also found an improvement with delta features and additional dependencies, but only when the edges were added discriminatively.

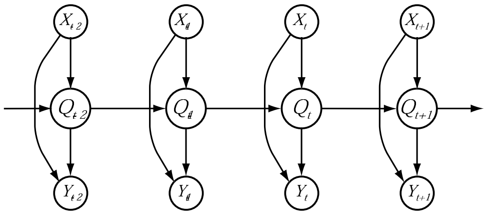


FIGURE 20. *An input-output HMM.* $X_{1:T}$ *the input is transformed via integration over a Markov chain* $Q_{1:T}$ *into the output* $Y_{1:T}$*.*

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 to describe some of them. While each of these models are quite different from one another, they can all be described with only simple modifications of an underlying graph structure.

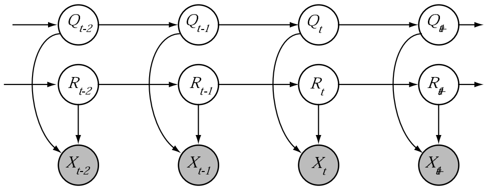


FIGURE 21. *A factorial HMM where there are multiple hidden Markov chains.*

The first example presented is a factorial HMM [67]. 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 21). 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 [90]. Alternatively, one chain could represent the speech to be recognized and the other chain could represent confounding background speech [151, 152][7], or the two chains might each represent two underlying concurrent and independent sub-processes governing the realization of the observation vectors [61, 155, 108]. Such factored hidden state representations have also been called HMM decomposition [151, 152] in the past.

One can imagine many modifications of this basic structure, where edges 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 [131, 132] or to represent a coupling between phonetic and articulatory information [167, 145].

---

[7] 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 [64].

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. 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 can have time complexity $O(TMK^{M+1})$ [67]. 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.

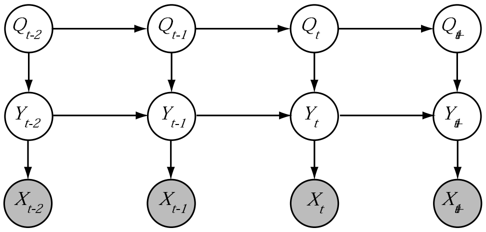


FIGURE 22. *The GM corresponding to a switching Kalman filter (SKM). The Q variables are discrete, but the Y and X variables are continuous. In the standard SKM, the implementations between continuous variables are linear Gaussian, other implementations can be used as well and have been applied to the 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 22. A version of this structure, applied to ASR, has been called a hidden dynamic model [125]. 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 [120]. In such cases, 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:

$$\begin{aligned} &p(X_{1:T} = x_{1:T}) \qquad (4.1) \\ &= \sum_{\tau} \sum_{q_{1:\tau}} \sum_{\ell_{1:\tau}} \prod_{i=1}^{\tau} p(x_{t(i,1)}, x_{t(i,2)}, \ldots, x_{t(i,\ell_i)}, \ell_i | q_i, \tau) p(q_i | q_{i-1}, \tau) p(\tau) \end{aligned}$$

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 $p(x|q)$. One of the first segment models [121] is a generalization that 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 [63]

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

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 [41, 42], the mean trajectory within a segment is described by a polynomial function over time. Equation 4.1 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 [120].

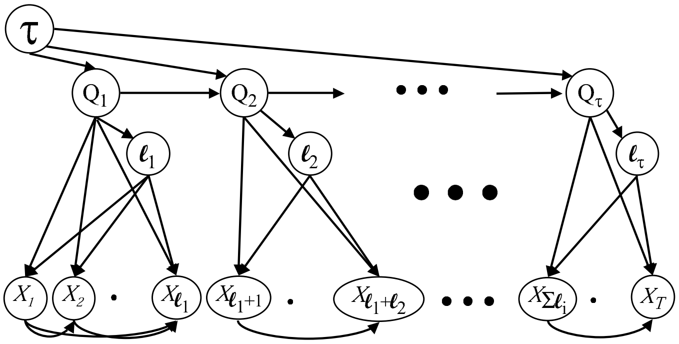


Figure 23. *A Segment model viewed as a GM.*

One can view a segment model as a GM as shown in Figure 23. A single hidden variable τ is shown that determines the number of segments. 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 explores several of them.

Perhaps the easiest way to use a graphical model for speech recognition is to start with the HMM graph given in Figure 16, and extend it with either additional edges or additional variables. In the former case, edges can be added between the hidden variables [43, 13] or between observed variables [157, 23, 14]. A crucial issue is how should the edges be added, as mentioned below. In the latter case, a variable might indicate a condition such as noise level or quality, gender, vocal tract length, speaking mode, prosody, pitch, pronunciation, channel quality, microphone type, and so on. The variables might be observed during training (when the condition is known), and hidden during testing (when the condition can be unknown).

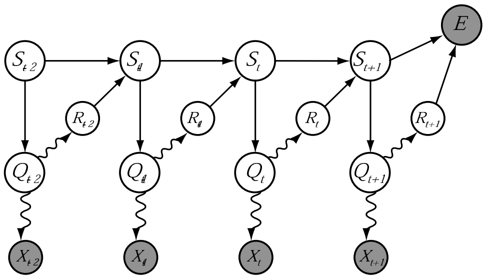


FIGURE 24. *A BN used to explicitly represent parameter tying. In this figure, the straight edges correspond to deterministic implementations and the rippled edges correspond to stochastic implementations.*

In each case, the number of parameters of the system will typically increase — in the worst of cases, the number of parameters will increase by a factor equal to the number of different conditions.

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 [167]. The simplest of cases is shown in Figure 24, 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 that 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 that 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 [167, 13, 47].

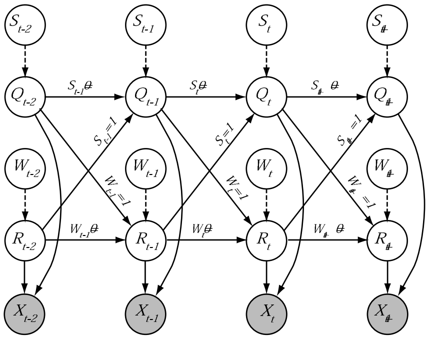


FIGURE 25. *A mixed-memory hidden Markov model. The dashed edges indicate that the S and the W nodes are switching parents.*

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 [142]. Viewed as a GM as in Figure 25, this model uses an additional hidden variable for each time frame and chain. Each normal hidden variables possesses an additional switching parent (as depicted by dotted edges in the figure, and as described in Section 2.2). 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:

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

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 [119].

A Buried Markov model (BMM) [16, 15, 14] 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

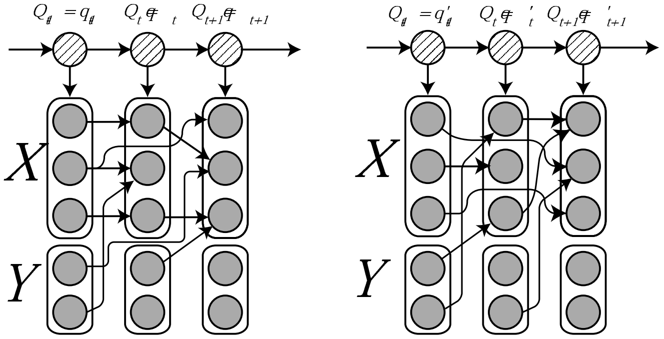


FIGURE 26. *A Buried Markov Model (BMM) with two hidden Markov chain assignments, $Q_{1:T} = q_{1:T}$ on the left, and $Q_{1:T} = q'_{1:T}$ on the right.*

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 $I(X_{t-1}; X_t | Q_t)$ [33]. 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 [16, 47, 166, 47]. 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 can be equivalent to optimizing the posterior probability of the class Q [16]. 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) [13]. 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, psycho-acoustics, 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 Theory*, 46:325–343, March 2000.

[3] T. Anastasakos, J. McDonough, R. Schwartz, and J Makhoul. A compact model

for speaker adaptive training. In *Proc. Int. Conf. on Spoken Language Processing*, 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*, 3:213–251, 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. What hmms can do. Technical Report UWEETR-2002-003, University of Washington, Dept. of EE, 2002.

[12] 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.

[13] J. Bilmes and G. Zweig. The Graphical Models Toolkit: An open source software system for speech and time-series processing. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2002.

[14] J.A. Bilmes. Data-driven extensions to HMM statistical dependencies. In *Proc. Int. Conf. on Spoken Language Processing*, Sidney, Australia, December 1998.

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

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

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

[18] 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.

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

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

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

[22] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, 1984.

[23] P.F. Brown. *The Acoustic Modeling Problem in Automatic Speech Recognition.* PhD thesis, Carnegie Mellon University, 1987.

[24] 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.

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

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

[27] R. Chellappa and A. Jain, editors. *Markov Random Fields: Theory and Appli-*

cation. Academic Press, 1993.

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

[29] 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.

[30] D.M. Chickering. *Learning from Data: Artificial Intelligence and Statistics*, chapter Learning Bayesian networks is NP-complete, pages 121–130. Springer-Verlag, 1996.

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

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

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

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

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

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

[37] A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41(1):1–31, 1989.

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

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

[40] 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.

[41] 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.

[42] 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.

[43] M. Deviren and K. Daoudi. Structure learning of dynamic bayesian networks in speech recognition. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 2001.

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

[45] E. Eide. Automatic modeling of pronunciation variations. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 6th, 1999.

[46] 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, 1982.

[47] J. Bilmes et al. Discriminatively structured graphical models for speech recognition: JHU-WS-2001 final workshop report. Technical report, CLSP, Johns Hopkins University, Baltimore MD, 2001.

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

[49] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Ann.*

Eugen., 7:179–188, 1936.

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

[51] V. Fontaine, C. Ris, and J.M.Boite. Nonlinear discriminant analysis for improved speech recognition. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 5th, pages 2071–2074, 1997.

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

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

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

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

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

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

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

[59] 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.

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

[61] 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.

[62] 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.

[63] M.J.F. Gales and S.J. Young. Segmental hidden Markov models. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 3rd, pages 1579–1582, 1993.

[64] 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.

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

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

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

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

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

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

[71] 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.

[72] A.K. Halberstadt and J.R. Glass. Heterogeneous measurements and multiple classifiers for speech recognition. In *Proc. Int. Conf. on Spoken Language Processing*, pages 995–998, 1998.

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

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

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

[76] 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.

[77] 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.

[78] 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.

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

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

[81] 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.

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

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

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

[85] M.I. Jordan and C. M. Bishop, editors. *An Introduction to Graphical Models.* to be published, 200x.

[86] 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.

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

[88] 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.

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

[90] 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.

[91] 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.

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

[93] 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.

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

[95] K. Kirchhoff. Combining acoustic and articulatory information for speech recognition in noisy and reverberant environments. In *Proceedings of the Interna-*

tional Conference on Spoken Language Processing, 1998.

[96] 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.

[97] 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.

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

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

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

[101] 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.

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

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

[104] 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.

[105] 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.

[106] 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.

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

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

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

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

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

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

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

[114] 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.

[115] M. Meilă. *Learning with Mixtures of Trees*. PhD thesis, MIT, 1999.

[116] 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.

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

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

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

[120] 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.

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

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

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

[124] 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.

[125] 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.

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

[127] 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.

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

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

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

[131] 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.

[132] 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.

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

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

[135] R. T. Rockafellar. *Convex Analysis.* Princeton, 1970.

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

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

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

[139] D. B. Rowe. *Multivariate Bayesian Statistics: Models for Source Separation adn Signal Unmixing.* CRC Press, Boca Raton, FL, 2002.

[140] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11:305–345, 1999.

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

[142] L.K. Saul and M.I. Jordan. Mixed memory markov models: Decomposing com-

plex stochastic processes as mixtures of simpler ones. *Machine Learning*, 1999.

[143] 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.

[144] 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.

[145] T. Stephenson, H. Bourlard, S. Bengio, and A. Morris. Automatic speech recognition using dynamic bayesian networks with both acoustic and articulatory variables. In *Proc. Int. Conf. on Spoken Language Processing*, pages 951–954, Beijing, China, 2000.

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

[147] M.E. Tipping and C.M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

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

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

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

[151] 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.

[152] A.P. Varga and R.K. Moore. Simultaneous recognition of concurrent speech signals using hidden makov model decomposition. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 2nd, 1991.

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

[154] 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.

[155] 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.

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

[157] 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.

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

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

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

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

[162] 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.

[163] 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.

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

[165] 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.

[166] G. Zweig, J. Bilmes, T. Richardson, K. Filali, K. Livescu, P. Xu, K. Jackson, Y. Brandman, E. Sandness, E. Holtz, J. Torres, and B. Byrne. Structurally discriminative graphical models for automatic speech recognition — results from the 2001 Johns Hopkins summer workshop. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2002.

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