ACADEMIC PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Speech and Language 17 (2003) 173–193

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

Bayesian network structures and inference techniques for automatic speech recognition

Geoffrey Zweig *

IBM, T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

Received 10 December 2001; received in revised form 20 November 2002; accepted 12 December 2002

Abstract

This paper describes the theory and implementation of Bayesian networks in the context of automatic speech recognition. Bayesian networks provide a succinct and expressive graphical language for factoring joint probability distributions, and we begin by presenting the structures that are appropriate for doing speech recognition training and decoding. This approach is notable because it expresses all the details of a speech recognition system in a uniform way using only the concepts of random variables and conditional probabilities. A powerful set of computational routines complements the representational utility of Bayesian networks, and the second part of this paper describes these algorithms in detail. We present a novel view of inference in general networks – where inference is done via a change-of-variables that renders the network tree-structured and amenable to a very simple form of inference. We present the technique in terms of straightforward dynamic programming recursions analogous to HMM α–β computation, and then extend it to handle deterministic constraints amongst variables in an extremely efficient manner. The paper concludes with a sequence of experimental results that show the range of effects that can be modeled, and that significant reductions in error-rate can be expected from intelligently factored state representations.

1. Introduction

Over the years, state-of-the-art speech recognition systems have grown in sophistication as code is added that addresses more and more of the phenomena that characterize natural speech in realistic environments. These phenomena include gender and age differences, pronunciation variability, differences in articulation, microphone and channel variability, and ambient noise.

* Tel.: +1-914-945-1204; fax: +1-914-945-4490.
E-mail address: gzweig@us.ibm.com.

0885-2308/03/$ - see front matter
doi:10.1016/S0885-2308(03)00007-X

In some cases, current techniques address the underlying issues fairly directly, as in vocal-tract length normalization (VTLN) (Zhan & Waibel, 1997), which attempts to compensate for variability in vocal-tract length, or parallel model combination PMC (Varga & Moore, 1990; Gales & Young, 1992), which builds separate noise and speech models to separate out the effects of ambient noise. In other cases, a technique will address a wide and undefined range of phenomena simultaneously, as with likelihood-increasing transforms such as maximum likelihood linear regression (MLLR) (Leggeter & Woodland, 1995), feature-space MLLR (FMLLR) (Gales, 1998), or discriminant transforms such as LDA. Typically, systems that incorporate the full range of methods are quite effective, as in current broadcast news (Eide et al., 2000) or Switchboard (Hain, Woodland, Evermann, & Povey, 2000) systems, but also significantly complex because the techniques were added in an evolutionary fashion over a long period of time.

The Bayesian network formalism offers an interesting and timely set of methods for addressing many of these issues in a highly systematic and unified way. The main idea of Bayesian networks as applied to speech recognition is to use a graphical structure to represent a concrete yet arbitrary set of variables within each time frame, and to specify an arbitrary factorization of the joint probability distribution over those variables. Because arbitrary sets of variables can be modeled, the formalism is ideally suited for representing detailed models of speech production and perception such as PMC, VTLN, probabilistic state classification (Luo & Jelinek, 1999), and articulatory models (Deng & Erler, 1992; Erler & Deng, 1993; Zweig, 1998; Richardson, Bilmes, & Diorio, 2000).

The application of Bayesian networks to speech recognition is best understood in terms of the broad categories of representational and computational power. As we will see in the following sections, what makes the formalism so unifying is that it reduces all phenomena to the common language of random variables and conditional probabilities. Hand-in-hand with this language is a collection of algorithms for computing with models described in the language. Once the algorithmic apparatus is in place, the same language is used to express everything from pronunciation dictionaries and trigram language models (Zweig, 1998) to sparse Gaussian covariances (Bilmes, 2000); and the same code is used to optimize the parameters for these seemingly disparate models, and to decode with them.

1.1. Bayesian networks as representational tools

A Bayesian network specifies a set of variables and a factorization of the joint probability distribution in a very straightforward way. The network is defined by a directed acyclic graph or DAG, in which the nodes represent variables and the edges induce a factorization of the joint probability distribution. Specifically, let the variables in the network be denoted by capital letters, e.g., X_i, and specific values by lower case letters, e.g., x_i. Further, suppose that the immediate predecessors or parents of a variable X_i are $\mathbf{X}_{\pi(i)}$ and that the values of the parents are $\mathbf{x}_{\pi(i)}$. Then the joint distribution is factored as

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \prod_i P(X_i = x_i | \mathbf{X}_{\pi(i)} = \mathbf{x}_{\pi(i)}).$$

In general, we will refer to a variable and its immediate predecessors as a family **f** in the graph. Fig. 1 illustrates a simple Bayesian network over five variables. As indicated by the graph topology, the joint distribution is given by

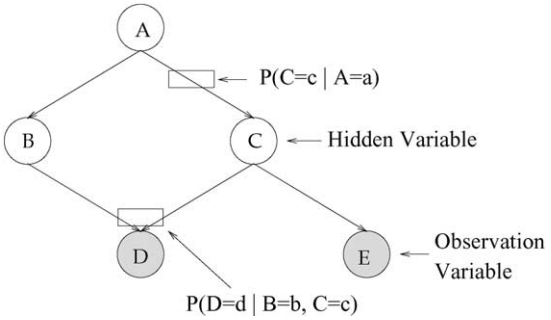

Fig. 1. A Bayesian network with five variables. Variables with know values are shaded. Conditional probability functions (indicated by boxes) are associated with each variable and used to return numerical values for conditional probabilities.

$$P(A=a,B=b,C=c,D=d,E=e) = P(A=a)P(B=b|A=a)P(C=c|A=a)P(D=d|B=b,C=c)P(E=e|C=c).$$

Fig. 1 also introduces the distinction between observed variables or ''observations'', whose value is unambiguously know (and which are shaded in the figure), and hidden variables whose values are unknown. We will refer to the set of hidden variables as **h** and observed variables as **o**. In the case that hidden variables are present, one may still reason in concrete terms by assuming various assignments of values to the hidden variables to obtain concrete data cases. The concept of concrete variable assignments is quite important, as it is the key to understanding the inference algorithms. As we shall see, decoding speech utterances will proceed by finding the single likeliest assignment to all the hidden variables in a network, and training will involve summing over all possible variable assignments.

In order to get numerical values for specific assignments of values to the variables, it is necessary to define concrete conditional probability functions, and here again there is a wide degree of flexibility. Some of the typical functions include tabular representations for discrete variables, and Gaussian distributions for real-valued variables. The set of operations that can be done with a graph depends on the types of variables in the graph, and the conditional probability functions that are used. In particular, the presence of hidden real-valued variables significantly complicates the associated algorithms and in some cases makes them computationally intractable. However, in speech recognition, continuous variables most commonly occur as observations, and in this paper we will assume that continuous variables are always observed. In this case, the structure of the inference algorithms is unchanged from the simplest fully discrete case.

1.2. Bayesian networks as computational tools

To be truly useful, it must be possible to compute with a representational framework, and here Bayesian networks are powered by a set of very general and yet efficient algorithms for computing the quantities that are relevant to speech recognition. Specifically, there are three main computational tasks that can be performed:

1. Computation of the probability of the observed variable values: $P(\mathbf{o}) = \sum_{\mathbf{h}} P(\mathbf{o},\mathbf{h})$.
2. Computation of the likeliest hidden variable values: $\text{argmax}_{\mathbf{h}} P(\mathbf{o},\mathbf{h})$.
3. Parameter estimation from a set of observations $\{\mathbf{o}_k\}$ via EM: $\text{argmax}_{\theta} \prod_k P(\mathbf{o}_k|\theta)$.

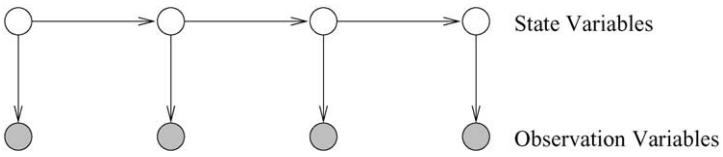

Fig. 2. A Bayesian network representation of the simplest HMM.

With the assumption of observed continuous variables, the process of finding the single likeliest assignment does not depend at all on the form of the conditional probability functions that is used. Similarly, the process of summing over all possible hidden variable assignments to obtain marginal probabilities is insensitive to the local distributions; however, in order to use the the parameter optimization routines, it must be possible to define an EM update rule for conditional probabilities represented in that form.

1.3. Organization

The remainder of this paper will focus in turn on the representational and computational aspects of Bayesian networks. Section 2 focuses on the representational, and presents the Bayesian network structures that are necessary to represent the components of an isolated word speech recognizer, or for rescoring N-best lists. In contrast to other comparisons of Bayesian networks with hidden Markov models, e.g. (Smyth, Heckerman, & Jordan, 1996), we make explicit all the detail required to capture the a priori knowledge, such as baseform sequences and parameter tying, that is present in recognition systems. Section 3 shifts to the computational, and presents the inference algorithms that enable training and decoding. In contrast to other descriptions of inference, that of Section 3 is tailored to efficiently handling speech recognition structures (which are characterized by deterministic relationships between many of the variables), and cast in terms of dynamic programming recursions that are direct extensions of the HMM recursions. Section 4 presents a set of experimental results for a Bayesian network system applied to an isolated word recognition task, and we summarize in Section 5.

2. Bayesian network structures for ASR

2.1. HMMs and Bayesian networks

Since the most commonly used model for speech recognition is the hidden Markov model (HMM) (Baker, 1975; Jelinek, 1997; Rabiner & Juang, 1993), we begin by relating HMMs to Bayesian networks. It has long been realized that the HMM is a special case of the more general class of dynamic Bayesian networks (Smyth et al., 1996), and Fig. 2 illustrates the representation of the simplest possible HMM.

Recall that in the classical definition (Rabiner & Juang, 1993), an HMM consists of:

- The specification of a number of states.
- An initial state distribution π.

- A state transition matrix A, where A_{ij} is the probability of transitioning from state i to j between successive observations.
- An observation function $b(i,t)$ that specifies the probability of seeing the observed acoustics at time t given that the system is in state i.

In this formulation, the joint probability of a state sequence $s_1, s_2, \ldots, s_T$ and observation sequence $o_1, o_2, \ldots, o_T$ is given by $\pi_{s_1} \prod_{i=1}^{T-1} A_{s_i s_{i+1}} \prod_{i=1}^{T} b(s_i, i)$. In the case that the state sequence or alignment is not known, the marginal probability of the observations can still be computed, either by enumerating all possible state sequences and summing the corresponding joint probabilities, or via dynamic programming recursions. Similarly, the single likeliest state sequence can be computed.

Fig. 2 shows the simplest Bayesian network representation of an HMM. It is a repeating model, in which each time frame has two variables: one whose value represents the value of the state at that time, and one that represents the value of the observation. The conditioning arrows indicate that the probability of seeing a particular state at time t is conditioned on the value of the state at time $t-1$, and the actual numerical value of this probability reflects the transition probability between the associated states in the HMM. The observation variable at each frame is conditioned on the state variable in the same frame, and the value of $P(o_t|s_t)$ reflects the output probabilities of the HMM.

One important thing to note about the Bayesian network representation is that it is explicit about time: each time frame gets its own separate segment in the model. In Fig. 2, there are exactly four time-slices represented, and to represent a longer time series would require a graph with more segments. This is in significant contrast to the HMM, which has no inherent mechanism for representing time. Instead, in the HMM framework, time is represented in the auxiliary data structures used for specific computations – the state transition matrix of an HMM does not have any reference to time.

Fig. 3 makes this more explicit. At the top of this figure is an HMM that represents the word "digit". There are five states (unshaded circles) representing the different sounds in the word, and a dummy initial and final state. The arcs in the HMM represent possible transitions, and *not* conditioning relationships as in the Bayesian network. This is a "left-to-right" HMM in which it is only possible to stay in the same state, or move forward in the state sequence. Because of self-loops, this kind of representation need not be explicit about time: it can represent 100 frame occurrences of the word "digit" as easily as 10 or 1000 frame occurrences. Of course, actual computations must be specific about time, and the temporal aspect is introduced into an HMM via the notion of a computational grid or its equivalent. This is shown at the upper right for a seven frame occurrence of the word "digit". The horizontal axis represents time; the vertical axis represents HMM-state, and a path from the lower-left corner of the grid to the upper-right corner represents an explicit path through the states of the HMM. In this case, the first frame is aligned to the /d/ state, the second and third frames are aligned to the /ih/ state, and so forth. Although the structure of the HMM is defined by the graph at the left, computations on the HMM are defined with reference to the temporally explicit grid.

The Bayesian network representation of the same utterance is shown at the bottom of Fig. 3. This is a temporally explicit structure with seven repeated segments, one for each of the time-slices. The assignment to the state variables of this model corresponds to the same HMM path that is represented in the computational grid. Note that different paths through the grid will correspond to different assignments of values to the state variables in the Bayesian network. Whereas computation in the HMM typically involves summing or maximizing over all *paths*,

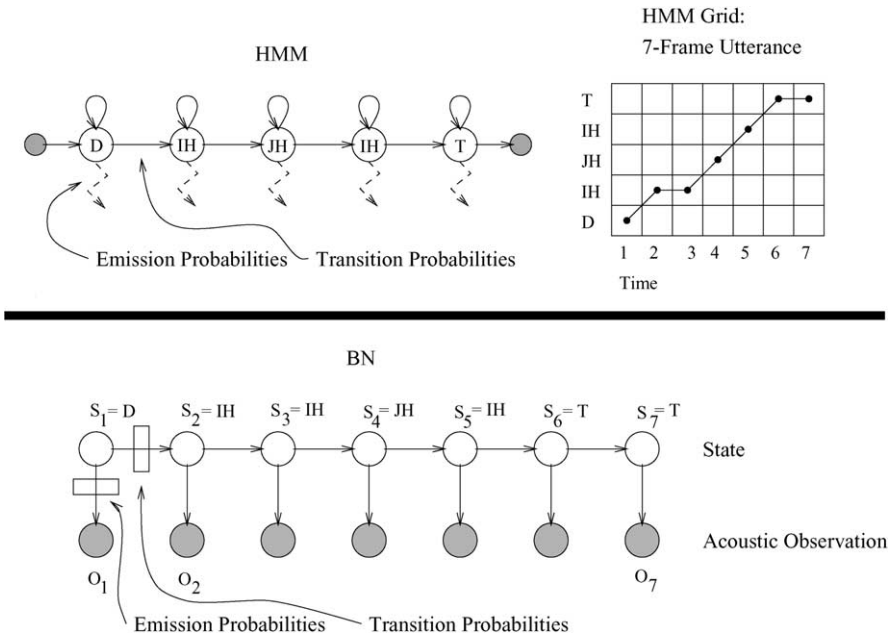

Fig. 3. Comparison of HMM and Bayesian network. The dashed lines in the HMM represent the acoustic emissions that occur at each time frame.

computation in the Bayesian network typically involves summing or maximizing over all possible *assignments to the hidden variables*. The analogy between a path in an HMM and an assignment of values to the hidden variables in a Bayesian network is quite important and should be kept in mind at all times.

Although the Bayesian network in Fig. 3 superficially represents the information in an HMM, closer consideration reveals that in fact there is some extra information that is typically present that this structure does not account for. One example of this is that in practice, the utterances that are associated with an HMM represent complete examples of words. A specific recording of the word "digit" will start with the /d/ sound and end with the /t/ sound; i.e., the whole word will be spoken. This extra piece of information is not captured in the previous Bayesian network. To see this, suppose the value /d/ is assigned to every occurrence of the state variable. (This will actually happen in the course of inference, either explicitly or implicitly depending on the algorithm used.) This concrete assignment will have some probability (gotten by multiplying together many instances of the self-loop probability for /d/ along with the associated observation probabilities), and this probability will not be zero. Yet this assignment violates our prior knowledge that a complete occurrence of "digit" has been spoken.

In the HMM framework, this corresponds to a path that does not end in the upper-right corner of the computational grid, and in typical systems the problem is solved by treating the upper-right corner in a special way. (Almost as if in addition to the vector π of state-priors there was a vector of allowable final states.) In the Bayesian network framework, prior knowledge such as this must be encoded in the representation itself: the inference algorithms are pure in the sense that they really do sum or maximize over all possible variable assignments.

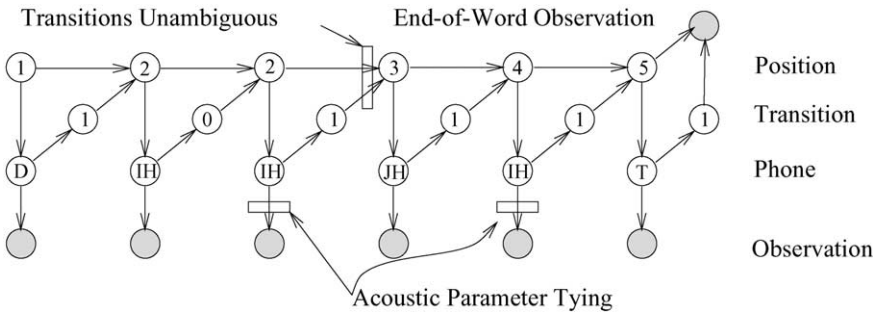

Fig. 4. A Bayesian network representation of a typical speech recognition HMM.

Another, more subtle, problem concerns the use of time-invariant conditional probabilities. Consider the "digit" example. Because the first occurrence of the phone /ih/ is followed by /jh/, it must be the case that $P(S_t = /jh/|S_{t-1} = /ih/) > 0$ – so that the transition is possible – and $P(S_t = /t/|S_{t-1} = /ih/) = 0$ – so that the transition is not skipped. However, because the second occurrence of /ih/ is followed by /t/, it must be the case that $P(S_t = /t/|S_{t-1} = /ih/) > 0$ and $P(S_t = /jh/|S_{t-1} = /ih/) = 0$ which is in direct contradiction to the previous requirements.

One way of solving this problem is to distinguish between the first and second occurrences of /ih/: to define /ih1/ and /ih2/. This is not satisfactory, however, because we may desire that the two share the same output probabilities. This problem is compounded when we consider instances of multiple words. For example, if the word "fit" – /f/ /ih/ /t/ – occurs in the database, should its /ih/ be the same as /ih1/ or /ih2/?

Similar to its handling of word-end constraints, the Bayesian network formalism solves the parameter tying problem via an explicit structural representation. We stress that HMM systems do of course implement parameter tying, but this is typically done with some sort of auxiliary file or table, whereas in a Bayesian network system it is expressed in terms of the common language of random variables and conditional probabilities. A Bayesian network that incorporates the word-end and parameter tying constraints that are appropriate to a practical application is shown in Fig. 4.

The main aspect of this representation is that there is now an explicit variable representing the position in the underlying HMM. The position within the HMM is distinct from the phone labeling the position, which is explicitly represented by a second set of variables. The tying problem is solved by mapping from position to phone via a conditional probability distribution. In the example of Fig. 4, positions 2 and 4 both map to /ih/. The probability distribution that will be used to model the acoustic observations will be the same in both cases. Time invariance is achieved via an explicit transition variable, which is conditioned on the phone. Either there is a transition or not, and the probability depends only on the phone. The position is a function of the previous position and the previous transition value: if the transition value was 0, then the position remains unchanged; otherwise it increments by one. These relationships can be straightforwardly encoded as conditional probabilities, ensuring representational consistency.

In this representation, the "end-of-word" observation is assigned an arbitrary value of 1, and ensures that all variable assignments that have non-zero probability must end on a transition out of the final position. This is done by conditioning this observation on the state variable, and setting its conditional probability distribution so that the probability of its observed value is 0

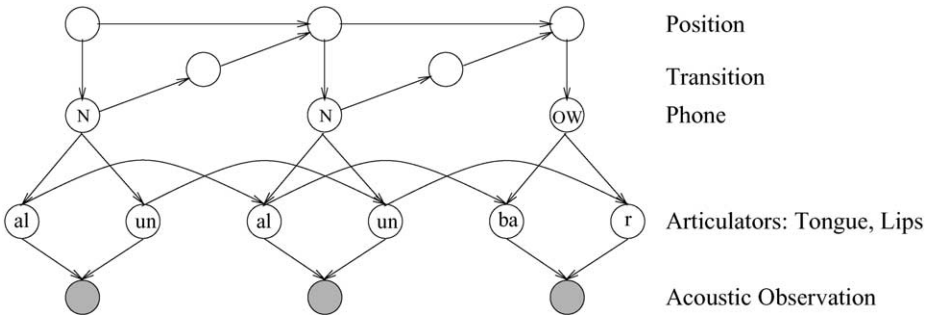

Fig. 5. An articulatory Bayesian network. Tongue and lips are explicitly represented, and values are assigned that are characteristic of an utterance of the word "no". The tongue begins in the alveolar position (al) and moves to the back of the mouth (ba). The lips start unrounded (un) and become rounded (r).

unless the position variable has the value of the last position. Similarly, its probability is 0 unless the transition value is 1. This ensures assignments in conformance with the usual HMM convention that all paths end in a transition out of the final emitting state.

In this model, the conditional probabilities need only be specified for the first occurrence of each kind of variable, and then can be shared by all subsequent occurrences of analogous variables. Thus, we have achieved the goal of making the conditional probabilities time-invariant. However, it is important to note that the probabilities *do* depend on the specific utterance being processed. In this model, the mapping from position to phone changes from utterance to utterance, as does the final position. Thus an implementation must support some mechanism for representing and reading in conditional probabilities on an utterance-by-utterance basis. This is analogous to reading in word-graphs, lattices, or transcripts on an utterance-by-utterance basis in an HMM system. The problem arises during training, when it is necessary to enforce known word sequences. In decoding, a fully utterance-independent network is used.

A final nuance of this model is that some of the conditional probability relationships are in fact deterministic, and not subject to parameter estimation. For example, the distribution controlling the position variable encodes a simple logical relationship: if the transition-parent is 0, then $Position_t = Position_{t-1}$; otherwise, $Position_t = Position_{t-1} + 1$. Efficiently representing deterministic relationships, and exploiting them in inference, is important for an efficient implementation of a Bayesian network system.

The Bayesian network in Fig. 4 is useful in a variety of tasks:

- Parameter estimation when the exact sequence of words and therefore phones, including silences, is known.
- Finding the single best alignment of a sequence of frames to a known sequence of words and phones.
- Computing the probability of acoustic observations, given a known word sequence – which is useful for rescoring the *n*-best hypotheses of other recognition systems.

Extensions to this basic structure can be made to make a continuous (rather than isolated) word decoder. Similarly, by adding extra variables, a variety of phenomena ranging from noise to coarticulation can be modeled. Fig. 5 illustrates this with an articulatory model, which has an additional layer of articulatory variables intermediate between the phone and observation

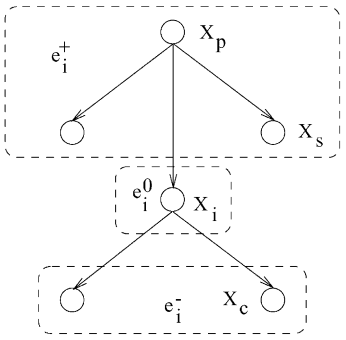

Fig. 6. A tree of variables.

variables to capture the fact that speech is produced in an articulatory process. A variety of other structures can be found in Zweig (1998).

3. Inference

In this section, we turn from representation to computation, and outline the procedures for computing with Bayesian networks. There are three main computational tasks that can be performed with the standard algorithms:

1. Computation of the probability of the observed variable values: $P(\mathbf{o}) = \sum_{\mathbf{h}} P(\mathbf{o}, \mathbf{h})$.
2. Computation of the likeliest hidden variable values: $\text{argmax}_{\mathbf{h}} P(\mathbf{o}, \mathbf{h})$.
3. Parameter estimation from a set of observations $\{\mathbf{o}_k\}$ via EM: $\text{argmax}_{\theta} \prod_k P(\mathbf{o}_k|\theta)$.

In general, inference in Bayesian networks is NP-hard (Cooper, 1990), but in practice many networks are tractable.

In the following sections, we present an inference technique for computing these quantities that is derived from Peot and Shachter (1991) and similar to Pearl (1988), Jensen, Lauritzen, and Olesen (1990), Spiegelhalter and Lauritzen (1990). However, it extends these methods by handling deterministic variables in a very efficient manner, and is cast purely in terms of simple dynamic programming recursions analogous to α–β computation in HMMs. The method is extremely straightforward in the case of tree-structured networks, and our presentation differs from others by casting the general case as a change-of-variables in which a complex network is transformed into an equivalent tree-structured network, where the original inference method is applied. We present a minimal set of conditions that must be met for this transformation to be valid.

3.1. Inference on trees

The simplest case is when the Bayesian network is tree structured, i.e., there is no path that leads from a variable back to itself, even if directionality is ignored. In this case, the runtime is proportional to the number of variables in graph, and the actual inference procedure is quite straightforward.

To see how tree-based inference works, consider the graph in Fig. 6. The variables that are relevant for the subsequent discussion are X_i: X_p, which is X_i's parent; X_s, which is its sibling; and X_c, which is its

Algorithm **Inference**()
for each variable X_i in postorder (Bottom-up pass)
  if X_i is a leaf
    $\lambda_j^i = 1,\ j \in CON(e_i^0);\ \lambda_j^i = 0,\ j \notin CON(e_i^0).$
  else
    $\lambda_j^i = \prod_{c \in children(X_i)} \sum_f \lambda_f^c P(X_c = f | X_i = j), j \in CON(e_i^0);\ \lambda_j^i = 0,\ j \notin CON(e_i^0).$

for each variable X_i in preorder (Top-down pass)
  if X_i is the root
    $\pi_j^i = P(X_i = j)$
  else
    let X_p be the parent of X_i
    $\pi_j^i = \sum_{v \in CON(e_p^0)} P(X_i = j | X_p = v) \pi_v^p \prod_{s \in siblings(X_i)} \sum_f \lambda_f^s P(X_s = f | X_p = v)$

Fig. 7. Inference in a tree. The preorder computation is analogous to the classical backwards algorithm the postorder computation is analogous to the classical forwards algorithm. In the simplest case where the tree is actually a chain, the two are identical, with the λs being identical to βs, and the πs to αs.

child. Note that a variable may have more than one child and/or sibling. In this graph, the values of some of the variables may be known, and we refer to these values as "evidence". Note that X_i partitions the remaining variables into two disjoint sets: the set of variables *rooted* in X_i, and all other variables. The set of evidence pertaining to the variables rooted in X_i is referred to as e_i^-, while the evidence pertaining to the variables not rooted in it is referred to as e_i^+. The evidence pertaining to X_i itself is referred to as e_i^0. We denote the set of values of X_i that are consistent with e_i^0 by $\mathrm{CON}(e_i^0)$. In the case at hand, where X_i is a single variable, this means that when X_i is hidden, $\mathrm{CON}(e_i^0)$ contains all its possible values, and when X_i is observed, $\mathrm{CON}(e_i^0)$ contains only its observed value. In section 3.2, we will consider the more general case of composite variables whose values range over the Cartesian product of a set of constituent variables. In this case, $\mathrm{CON}(e_i^0)$ refers to the set of cross-product values that are consistent with all known constituent values.

Note that the union of the evidence includes all the observations, and the probability of all the evidence and variable X_i taking value j is

$$P(e_i^+, e_i^-, e_i^0, X_i = j) = P(e_i^+, X_i = j) P(e_i^-, e_i^0 | X_i = j, e_i^+) = P(e_i^+, X_i = j) P(e_i^-, e_i^0 | X_i = j).$$

In the case that $X_i = j$ contradicts e_i^0, $P(e_i^-, e_i^0 | X_i = j)$ is zero.

This leads to a natural factorization, and in the inference procedure the following two key quantities will be calculated for each variable X_i:

- $\lambda_j^i = P(\mathbf{e}_i^-, \mathbf{e}_i^0 | X_i = j)$.
- $\pi_j^i = P(\mathbf{e}_i^+, X_i = j)$.

It follows from these definitions that for every variable X_i,

- $P(Observations) = \sum_j \lambda_j^i \pi_j^i$.
- $P(X_i = j | Observations) = \lambda_j^i \pi_j^i / \sum_j \lambda_j^i \pi_j^i$.

Fig. 7 presents an algorithm for computing these values. In an efficient implementation, it is useful to define τ_v^i as $\tau_v^i = \sum_f \lambda_f^i P(X_i = f | X_\mathrm{p} = v)$ and to cache it in the bottom-up pass (Peot & Shachter, 1991; Jensen et al., 1990; Hugin-Expert, 1995). Then the product over siblings in the original formula can be done with a single division. (If $\tau_v^i = 0$, then any term involving division

Algorithm **Viterbi** ()
for each variable X_i in postorder
  if X_i is a leaf
    $\lambda_j^{*i} = 1,\ j \in CON(e_i^0); \lambda_j^{*i} = 0,\ j \notin CON(e_i^0).$
  else
    $\lambda_j^{*i} = \prod_{c \in children(X_i)} \max_f \lambda_f^{*c} P(X_c = f | X_i = j),\ j \in CON(e_i^0); \lambda_j^{*i} = 0,\ j \notin CON(e_i^0).$

Fig. 8. Viterbi inference in a tree. This produces the probability of the likeliest variable assignments. The actual variable values can be recovered by storing the child-values that lead to each λ_j^{*i}.

by it can be also set to 0, i.e., ignored in the sum.) With this caching, and if we assume a tree with N variables each of cardinality k, the runtime is $O(Nk^2)$. It is easy to see that the runtime cannot be improved on: in general, there are $O(Nk^2)$ conditional probabilities associated with the network, so it takes that long just to read in all the parameters. We also note that even if observations are continuous, the fact that they are observed effectively eliminates any summing over their values.

To obtain the likeliest values for the variables, we denote an assignment to the hidden variables rooted in X_i as $\mathbf{x}_i^-$, and redefine the λ as follows:

$$\begin{aligned}\lambda_j^{*i} &= \text{Probability of the likeliest values for the variables rooted in } X_i \\ &= \max_{\mathbf{x}_i^-} P(e_i^-, e_i^0, \mathbf{x}_i^- | X_i = j).\end{aligned}$$

The recursions for computing the λ_j^{*i} are given in Fig. 8, and the actual values $\mathbf{x_i}$ can be obtained by storing the child-values that led to each λ^*. Before turning to general graphs, we note that the probabilities that are computed will typically be too small to represent with a floating point number. Therefore, numbers are represented by their logarithms, with suitable functions for addition and multiplication.

3.2. Inference in general graphs

The algorithm presented in section 7 works only in tree-structured graphs. In fact, with a change-of-variables, the same method can be used for general graphs. The idea is to define a new tree-structured network in terms of a new set of variables in such a way that the new network represents exactly the same joint probability distribution as the old network, and then to use the simple tree-inference algorithm. A detailed proof of the procedure appears in Zweig (1998), and only a summary is presented here. The main idea is to define new variables or cliques that represent sets of old variables. Trees constructed with these cliques can be shown to be computationally valid as long as

- each original variable occurs in a clique with its parents in the original graph;
- the original variables satisfy the running intersection property in the clique tree, i.e., if a variable occurs in two cliques, then it must occur in all the cliques connecting them.

Methods for constructing such trees can be found in Pearl (1988), Spiegelhalter and Lauritzen (1990), Zweig (1998), but anything that satisfies these properties will suffice.

The inference process can now be summarized: the variables in a clique tree represent subsets of the original variables. Each new clique variable can take a distinct value for every possible assignment of values to its members. Each variable in the original network is assigned to a single clique in which it occurs with its parents (when there are several such cliques, and an arbitrary

choice may be made). The inference procedure outlined in Fig. 7 works on clique trees with the following change-of-variables:

- $\mathbf{e}_i^0$ is the set of observed values for the evidence variables assigned to clique C_i;
- $\mathbf{e}_i^-$ is the set of observed values for the evidence variables assigned to cliques in the subtrees rooted in C_i;
- $\mathbf{e}_i^+$ is the set of observed values for all other evidence variables;
- $\lambda_j^i = P(\mathbf{e}_i^- | C_i = j)$;
- $\pi_j^i = P(\mathbf{e}_i^+, C_i = j)$;
- $\mathbf{F}_c$ is the set of variables assigned to C_c;
- $P(C_c = i | C_p = j) = 0$ if i and j imply inconsistent values for shared variables. Otherwise, if $\mathbf{F}_c \neq \emptyset$,
  - $P(C_c = i | C_p = j) = \prod_{X_k \in \mathbf{F}_c} P(x_k | Parents(X_k))$, with x_k and $Parents(X_k)$ implied by i, j;
  - $P(C_r = i) = \prod_{X_k \in \mathbf{F}_r} P(x_k | Parents(X_k))$, with x_k and $Parents(X_k)$ implied by i;
- any remaining conditional probability is defined to be 1.

3.3. EM

The λ and π quantities that are computed during inference can be used to do parameter estimation via EM. In the case of discrete conditional probability tables, the crux of the EM algorithm is extremely simple. It works in terms of the number of times families of variables are seen with specific values. If we let N_{ijk} be the number of times X_i is seen with value k and its parents have instantiation j, then we estimate θ_{ijk}, the probability that $X_i = k$ given that its parents have instantiation j, as $N_{ijk} / \sum_k N_{ijk}$ (Lauritzen, 1991; Heckerman, 1995). When both a variable and its parents have observed values, N_{ijk} is obtained simply by counting. More commonly, there are some unknown values in which case inference is necessary. The calculations can be summarized as follows.

Let C_i be the clique containing x_i and its parents. Let $\mathbf{V}_{jk}^i$ be the set of C_i's clique values corresponding to underlying variable assignments that include $X_i = k, Parents(X_i) = j$. Now, the expected value of N_{ijk} given the observations and current parameters can be found by summing over the appropriate clique values:

$$N_{ijk} = \sum_{w \in \mathbf{V}_{jk}^i} \frac{\lambda_w^i * \pi_w^i}{\sum_w \lambda_w^i * \pi_w^i}.$$

Estimating N_{ijk} from a collection of examples simply requires summing the individual estimates for each example. By maintaining the appropriate data structures, the counts for every family can be computed in a single sweep over the cliques. When the observations are real valued, this approach can be generalized, analogous to HMMs, to collect the sufficient statistics required for Gaussian re-estimation.

3.4. Implementation speedups

As with most theoretical frameworks, a number of tricks must be employed to make practical implementations perform well. In the case of Bayesian networks, two of the most important are the use of separators, and the careful handling of deterministic variables.

Separator variables represent the variables in $\{C_p \cap C_c\}$. The value of a separator's parent uniquely defines the separator's value, and the conditional probability of a separator taking its one allowed value given its parent's value is always 1. The use of separators can have a dramatic effect on running time; for example, suppose there is a network consisting of variables A, B, C and D, which take a, b, c and d values, respectively. In a clique tree consisting of the cliques $\{A, B, C\}$ and $\{B, C, D\}$, the introduction of a separator-variable representing $\{B, C\}$ reduces the work from $a * b * c * d$ to $a * b * c + b * c * d$.

The efficient handling of deterministic variables is also quite important: as we have seen, modeling word pronunciations with a Bayesian network requires the extensive use of deterministic variables. Unless the constraints imposed by deterministic variables are exploited, runtime increases because of a vast number of "0-probability" events that must be considered, and storage increases due to large numbers of 0s stored in tables. We note that even a statically compiled sparse representation will not in general be optimal, because even this stores distributions for parent-values that may be impossible *given the evidence*. Dynamically identifying possible values can take into account the evidence.

This can be done by making a preorder traversal of the tree and recursively identify the values that are legal – i.e., that have non-zero probability – for a child clique, given the just-computed legal values of the parent. In this manner, the realistic clique values can be identified, and the inference sums constrained. The extra cost involved is an additional pass over the tree.

In order to resolve the legal values in a single pass, it is necessary that the first time a deterministic variable appears in a clique, its parents also be present. Without this constraint, the (unique) value of a deterministic variable cannot necessarily be resolved when it is first encountered. This can be guaranteed with minor modifications to standard clique-tree building techniques (Jensen, Jensen, & Dittmer, 1994; Pearl, 1988; Lauritzen & Spiegelhalter, 1988; Zweig, 1998).

The bounds on inference runtime can be straightforwardly expressed. Let d_{C_i} denote the degree of clique i; i.e., the total number of edges incident on C_i including both incoming and outgoing edges. Let s_{C_i} denote the total number of possible values C_i can take. Once the possible clique values are enumerated, an examination of the inference loops reveals that the running time of the actual inference procedure is $O(\sum_{C_i \in Non\text{-}Separators} d_{C_i} s_{C_i})$.

This time bound is slightly more precise than, but in the worst case the same as, the the bound presented in Lauritzen and Spiegelhalter (1988). This bound cannot be improved on by any procedure in which each clique transmits information to all its neighbors. We conclude this section by noting that in practice, a well-tuned HMM system can be expected to be faster than a similarly tuned Bayesian network system. There is an inevitable overhead to be paid for the generality of operating on arbitrary graphs.

4. Speech recognition experiments

This section presents experimental results for a fully implemented Bayesian network speech recognition system. The task we address is a large-vocabulary multi-speaker database of isolated words. The database is challenging enough that results are significant, yet because it has isolated words, it avoids many issues that complicate continuous-speech ASR systems. In short, the

database is just complex enough to test some basic issues relating to the use of factored state representations in ASR.

4.1. Database

In these experiments, we use the Phonebook database (Pitrelli, Fong, Wong, Spitz, & Leung, 1995), a collection of words chosen to exhibit all the coarticulatory effects found in the English language. It therefore contains a wide variety of words, for example, "haymarket", "barleycorn", and "immobilizing". The utterances were collected over the telephone, and thus contain a variety of transmission distortions.

The utterances were collected from a group of American speakers who were "balanced for gender, and demographically representative of geographic location,..., income, age (over 18 years), education, and socio-economic status" (Pitrelli et al., 1995). Each speaker was asked to read 75 or 76 words, and the utterances were then screened for acceptable pronunciation.

4.2. Acoustic processing

The utterances were processed with relatively conventional acoustic processing. Initial and final silence was removed using the endpoints provided with the database. Then the utterances were divided into 25 ms windows and MFCCs were calculated. The analysis windows overlapped by 2/3.

Smoothed MFCC derivatives (Rabiner & Juang, 1993) were computed, and three data streams were created: one for the MFCCs, one for the derivatives, and one for the combination of C_0 and delta-C_0 (which were omitted from the first two streams). Mean cepstral-subtraction (Mammone, Zhang, & Ramachandran, 1996) was performed for cepstral coefficients $C_1 - C_{10}$, and speaker normalization (Lee, 1989) was done for C_0.

The data streams were vector-quantized to eight bits (256 values). The MFCCs and delta-MFCCs were quantized in separate codebooks. C_0 and delta-C_0 were quantized to four bits each, and then concatenated to form a single eight-bit stream.

4.3. Phonetic alphabets

Results are presented for Bayesian network models using both context-independent and context-dependent phonetic units. The Phonebook database provides phoneme-level transcriptions, and these formed the basis of both kinds of alphabet. To keep the number of parameters reasonable, and in common with other work (Dupont, Bourlard, Deroo, Fontaine, & Boite, 1997), we did not use the 3-way stress distinction for vowels. We used 41 basic phonemes, plus initial and final silence units.

In the case of context-independent units, i.e., simple phonemes, each phoneme was replaced by a k-state left-to-right phone model. Most of the experiments report results for 4-state phone models, which experimentation shows this to be a good number. In all cases, one-state models were used to represent silence.

To create a context-dependent alphabet, we used a variation on diphones (Schwartz, Klovstad, Makhoul, & Sorensen, 1980). The basic idea is to create two new units for each phoneme in a transcription: one for the initial part of the phoneme in the left-context of the preceding phoneme,

and one for the final part of the phoneme in the right-context of the following phoneme. Thus, for example, /*k ae t*/ becomes

(*sil* **k**)(**k** *ae*)(*k* **ae**)(**ae** *t*)(*ae* **t**)(**t** *sil*).

This scheme has the advantage of addressing both left and right contexts, like triphones, while only squaring – rather than cubing – the number of potential phonetic units.

To prevent overtraining, a context-dependent unit was used only if it occurred a threshold number of times in the training data. Units that did not meet this criterion were replaced with context-independent units. We used thresholds of 250 and 125, which resulted in alphabets with sizes of 336 and 666, respectively, including a full set of context-independent units. We found it beneficial to double the occurrence of each of the units in a context-dependent transcription, and so increase the minimum and expected state durations. Thus, the total number of phones is four times the original number of phonemes, the same number that results from four-state context-independent phoneme models.

In the following sections, we will present experimental results for networks in which an exta (beyond what is present in the basic HMM structure) or "auxiliary" variable is used in the observation modeling. It is important to realize that this auxiliary variable is also capable of conveying contextual information, such as noise or articulatory conditions. However, context as expressed with a hidden variable in a Bayesian network is significantly different from that expressed in a context-dependent alphabet. Context of the kind expressed in a context-dependent alphabet is based on an idealized and invariant pronunciation template; a word model based on context-dependent phones can be written down before ever seeing a sound wave, and therefore represents a-priori knowledge. The auxiliary-variable represents context as manifested in a specific utterance. Although we choose to use both a context-dependent alphabet and an auxiliary variable, both types of information could be encoded in a network with more variables.

4.4. Experimental procedure

4.4.1. Training, tuning and testing

The database was divided into separate portions for training, tuning, and testing as specified by (Dupont et al., 1997). This resulted in 19,421 training utterances, 7291 tuning utterances and 6598 test utterances, with no overlap between the speakers or words in any of the partitions. All decisions concerning network structures and alphabets were made by training a system on the training data, and testing it on the tuning data. Decisions were not made on the basis of performance on the actual test data. The words in the Phonebook vocabulary are divided into 75 and 76-word groups, and the recognition task consists of identifying each test word from among the other words in its subset.

4.4.2. Models tested

A baseline Bayesian network was constructed to emulate an unfactored HMM. Bayesian networks with one or more auxiliary state variables were then designed to answer the following questions:

1. What is the effect of modeling correlations between observations within a single timeslice? Specifically,

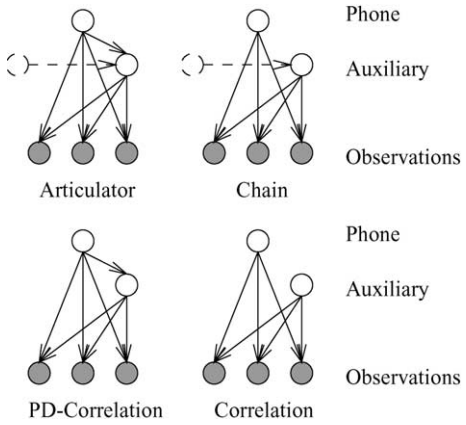

Fig. 9. The acoustic models for four of the network topologies tested. The dotted lines indicate conditioning on the previous frame.

(a) What is the effect of modeling these correlations in a phone-independent way? This question was addressed with the "Correlation" network of Fig. 9. This network is only capable of modeling intra-frame observation correlations in the most basic way.
(b) What is the effect of modeling these correlations in a phone-dependent way? This question was addressed with the phone-dependent "PD-Correlation" network of Fig. 9. This network can directly model phone-dependent intra-frame correlations among the acoustic features, and is comparable to using full-covariance rather than diagonal covariance Gaussians.

2. What is the effect of modeling temporal continuity? Specifically,
   (a) What is the effect of modeling temporal continuity in the auxiliary chain in a phone-independent way? This question was addressed with the "Chain" network of Fig. 9. This network results from the addition of temporal links to the auxiliary variable of the correlation network, and can directly represent phone-independent temporal correlations. The network was initialized to reflect continuity in the value of the auxiliary variable.
   (b) What is the effect of modeling temporal continuity in the auxiliary chain in a phone-dependent way? This was addressed with the "articulator" network of Fig. 9. In this network, the auxiliary variable depends on both the phonetic state and its own past value. This can directly represent phone-dependent articulatory target positions and inertial constraints. The network was initialized to reflect voicing.
3. How does the use of a context-dependent alphabet compare to context-modeling with an auxiliary variable? This was addressed by using a context-dependent alphabet in a network with no auxiliary variable. Additionally, we tested the combination of a context-dependent alphabet with an auxiliary variable.
4. What is the effect of increasing the number of values in the auxiliary chain, and how does increasing this number compare to increasing the number of auxiliary variables? This question was answered by making the proposed changes to the chain network.
5. What is the effect of using an unfactored state representation to represent the same process? To answer this question, a Bayesian network was used to emulate an HMM with an unfactored representation.
6. What is the effect of doing unsupervised clustering?

4.5. Results with a single auxiliary variable

Answers to the first three questions are presented in this section. Table 1 shows the word error-rates with the basic phoneme alphabet, for the network structures shown in Fig. 9. The results for the Bayesian networks with an auxiliary variable are consistently better than without one. A large improvement results simply from modeling within-frame correlations, but for both the Correlation and the PD-Correlation networks, a further improvement results from the addition of temporal-continuity links. The Articulator network provided the best performance. Note that because the observation variable could take many more values than the others, adding links between the others has little impact on the number of parameters.

In terms of absolute error-rates, these results compare favorably with those reported in Dupont et al. (1997). That paper reports an error-rate of 4.1% for an ANN-HMM hybrid using the Phonebook baseform transcriptions, and the same training and test sets. However, with phonetic transcriptions based on the CMU dictionary, rather than with the supplied transcriptions (Dupont et al., 1997), achieved results as low as 1.5%. Comparison with this work provides a useful check on the overall recognition rate, but it should be remembered that a vector-quantized system is being compared with a continuous-observation system. Thus differences are difficult to interpret.

4.5.1. Context-dependent alphabet

Error rates with the context-dependent alphabets are reported in Table 2, and improve significantly on the context-independent results. As discussed previously, context as represented in an alphabet is different from context as represented in a Bayesian network with an auxiliary variable. Therefore it makes sense to combine the two strategies, and this in fact produced the best

Table 1
Test set word error-rate for systems using the basic phoneme alphabet

Network	Parameters	Error rate
Baseline-HMM	127 k	4.8%
Correlation	254 k	3.7%
PD-Correlation	254 k	4.2%
Chain	254 k	3.6%
Articulator	255 k	3.4%

All the systems had slightly different numbers of parameters. The standard error is approximately 0.25%. Results from Zweig and Russell (1998).

Table 2
Test set word error-rates for systems using context-dependent alphabets

Network	Parameters	Error rate
CDA-HMM	257 k	3.2%
CDA-Articulator	515 k	2.7%
CDA-HMM	510 k	3.1%

The first two results use an alphabet with 336 units and the last result uses an alphabet with 666 units. The standard error is approximately 0.20%. Results from Zweig and Russell (1998).

Table 3
Test results with multi-valued and multi-chain auxiliary variables; the standard error is approximately 0.25%

Network	Parameters	Error rate
Binary-chain	191 k	4.1%
Trinary-chain	287 k	4.0%
Quaternary-chain	383 k	3.8%
Double-chain	383 k	3.6%

The double-chain network used binary variables, and thus had a total of four possible values.

overall results. Adding an extra auxiliary variable doubled the number of parameters (because there is now a separate distribution for each combination of phone and auxiliary value), but as the last line in Table 2 indicates, doubling this number by using a bigger alphabet is not as effective.

4.6. Results with two auxiliary variables

In order to evaluate the use of multiple auxiliary chains, we tested a network with two auxiliary variables. Both of the variables were binary. For comparison, a network with a single auxiliary variable with 3 and 4 values was tested. To cut down on memory usage and running time, and to save on the amount of disk-space needed to store conditional probabilities, 3-state phones were used in this set of experiments. The results are shown in Table 3.

These results indicate that increasing the amount of auxiliary state improves recognition performance. In addition, factoring the auxiliary state is beneficial.

4.7. Clustering results

This section presents results for a network doing unsupervised clustering. A binary-valued auxiliary variable was used in the chain structure, with the restriction that its value not change over the course of an utterance. This was enforced by using a stochastic auxiliary variable in the first timeslice, and then using a deterministic auxiliary variable from the second timeslice on to copy the value determined in the first frame. The network was trained as usual, and then during testing the likeliest value for the cluster variable was determined.

There are at least two dimensions along which one might expect clustering to occur: the type of speaker (e.g., male vs. female, adult vs. child), and the type of word (e.g., consonant-initial vs. vowel-initial). [1] The degree to which such clustering occurs can be measured by looking at the degree to which utterances with a particular characteristic are classified together in a single cluster. Fig. 10 shows the consistency with which utterances from a single speaker were classified together, and what would be expected at random. Clearly, utterances from individual speakers are being grouped together with high frequency. A similar plot results for the fraction of times that utterances of a single word (across multiple speakers) are classified together, indicating that clustering along both dimensions is present.

In terms of overall error-rate, the clustering technique did not do as well as the other augmented networks; the test-set error-rate of 4.5% was midway between the 4.8% rate of the baseline

[1] Thanks to Jeff Bilmes for pointing out the duality between speakers and words.

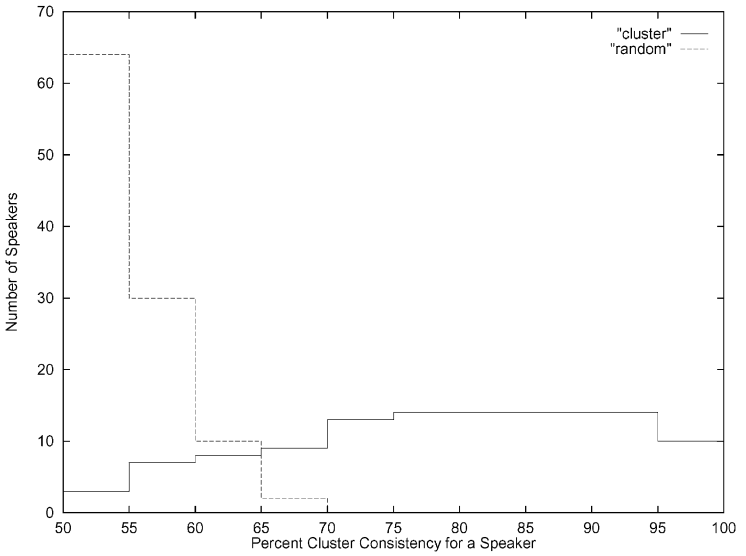

Fig. 10. The frequency with which utterances from a single speaker was assigned to the same cluster. For example, about 15 speakers had their utterances clustered together with 85% consistency. On average, there are 68 utterances per speaker. The dotted line shows the distribution resulting from a random assignment of speakers to clusters. Far more speakers show a low degree of cluster consistency.

network and the 3.6% score for the chain network. This is expected, since the cluster network has more state, and therefore modeling power, than the baseline network, but not as much expressiveness as the chain network, where the auxiliary variable can "flip-flop" between values.

It is possible to make sense of the value assigned to the cluster variable, both in terms of speaker-type and word-type. The mutual information between the cluster variable and the gender of the speaker is 0.24 bits, indicating a strong correlation between cluster and gender. To determine the word-characteristics associated with the two clusters, we examined the words that were very consistently assigned to a particular cluster. One of the clusters is characterized by words beginning in liquid consonants – for example, "landberg" and "lifeboat" – while the other is characterized by words ending in liquid consonants – for example, "mistrustful" and "unquestionable". The cluster with words beginning in liquid consonants also happens to be associated with female speakers; the cluster with terminal liquid consonants is associated with male speakers. Note, however, that since each word was spoken by approximately as many men as women, word-clustering comes at the expense of gender-clustering.

4.8. Discussion

In every case that an auxiliary variable was used, there was a performance increase. Moreover, increasing the modeling power of the network by increasing the amount of auxiliary state produced further improvements. This indicates that context modeling with auxiliary state information is an effective way of decreasing speech recognition error-rates.

The auxiliary variable was able to capture several different phenomena, ranging from simple correlations between the observations within a single frame to gender and word-specific

characteristics. In general, networks in which the auxiliary variables were linked across time did better than corresponding networks without temporal links. This is unsurprising because neither acoustic nor articulatory properties are expected to change rapidly over time.

A context-sensitive alphabet was an effective way of improving performance, but here too an auxiliary variable was beneficial. This makes sense because context-dependent alphabets encode prior knowledge about coarticulatory effects, but do not pay attention to the particulars of any specific utterance. An auxiliary variable has the ability to encode information on a case-by-case basis.

5. Conclusion

In this paper, we have seen that Bayesian networks can be used to model a wide variety of phenomena that occur in speech production and recognition. Using the two concepts of random variables and conditional probabilities, we are able to express many of the a priori constraints that exist in a recognizer, e.g., baseform spellings, as well as physiologically based phenomena such as coarticulation. In addition to expressive power, Bayesian networks are endowed with a set of general purpose algorithms that allow for parameter optimization and Viterbi decoding with arbitrary network structures. We have presented these algorithms in terms of straightforward dynamic programming recursions, and showed how to extend them to efficiently handle the deterministic constraints that often occur amongst variables in a speech recognition network.

References

Baker, J., 1975. The Dragon system – an overview. IEEE Transactions on Acoustics, Speech, and Signal Processing 23, 24–29.

Bilmes, J., 2000. Factored sparse inverse covariance matrices. In: ICASSP 2000.

Cooper, G.F., 1990. The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence 42, 393–405.

Deng, L., Erler, K., 1992. Structural design of hidden Markov model speech recognizer using multivalued phonetic features: comparison with segmental speech units. Journal of the Acoustical Society of America 92, 3058–3067.

Dupont, S., Bourlard, H., Deroo, O., Fontaine, J.-M., Boite, V., 1997. Hybrid HMM/ANN systems for training independent tasks: experiments on PhoneBook and related improvements. In: CASSP-97: 1997 International Conference on Acoustics, Speech, and Signal Processing. IEEE Computer Society Press, Los Alamitos, CA, pp. 1767–1770.

Eide, E., Maison, B., Chen, S., Olsen, P., Kanevsky, D., Gopinath, R.A., 2000. Ibm's 10xrt system in the, 1999 arpa broadcast news evaluation. In: ICSLP 2000.

Erler, K., Deng, L., 1993. Hidden Markov model representation of quantized articulatory features for speech recognition. Computer Speech and Language 7, 265–282.

Gales, M.J.F., 1998. Maximum likelihood linear transformations for hmm-based speech recognition. Computer Speech and Language, 12.

Gales, M.J.F., Young, S., 1992. An improved approach to the hidden Markov model decomposition of speech and noise. In: ICASSP-92, I-233–I-236.

Hain, T., Woodland, P.C., Evermann, G., Povey, D., 2000. The cu-htk march 2000 hub5e transcription system. In: Proceedings of the Speech Transcription Workshop 2000.

Heckerman, D., 1995. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington. Revised June 1996.

Hugin-Expert., 1995. HUGIN API Reference Manual. Hugin Expert A/S.
Jelinek, F., 1997. Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA.
Jensen, F., Jensen, F.V., Dittmer, S.L., 1994. From influence diagrams to junction trees. In: Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference. Morgan Kaufmann, Seattle, WA, pp. 367–373.
Jensen, F.V., Lauritzen, S.L., Olesen, K.G., 1990. Bayesian updating in causal probabilistic networks by local computations. Computational Statistics Quarterly 5, 269–282.
Lauritzen, S.L., 1991. The EM algorithm for graphical association models with missing data. Technical Report TR-91-05, Department of Statistics, Aalborg University.
Lauritzen, S.L., Spiegelhalter, D.J., 1988. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society B 50, 157–224.
Lee, K.-F., 1989. Automatic Speech Recognition: The Development of the SPHINX System. Kluwer Academic Publishers, Dordrecht, The Netherlands.
Leggeter, C., Woodland, P.C., 1995. Flexible speaker adaptation using maximum likelihood linear regression. In: Eurospeech-95.
Luo, X., Jelinek, F., 1999. Probabilistic classification of hmm states for large vocabulary continuous speech recognition. In: ICASSP 1999.
Mammone, R.J., Zhang, X., Ramachandran, R.P., 1996. Robust speaker recognition. IEEE Signal Processing Magazine, 58–71.
Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA.
Peot, M., Shachter, R., 1991. Fusion and propagation with multiple observations. Artificial Intelligence 48, 299–318.
Pitrelli, J., Fong, C., Wong, S., Spitz, J., Leung, H., 1995. Phonebook: A phonetically-rich isolated-word telephone-speech database. In: ICASSP-95: 1995 International Conference on Acoustics, Speech, and Signal Processing. IEEE Computer Society Press, Los Alamitos, CA, pp. 101–104.
Rabiner, L.R., Juang, B.-H., 1993. Fundamentals of Speech Recognition. Prentice-Hall, Englewood Cliffs, NJ.
Richardson, M., Bilmes, J., Diorio, C., 2000. Hidden-articulatory markov models for speech recognition. In: ICASSP-2000.
Schwartz, R., Klovstad, J., Makhoul, J., Sorensen, J., 1980. A preliminary design of a phonetic vocoder based on a diphone model. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 32–35.
Smyth, P., Heckerman, D., Jordan, M., 1996. Probabilistic independence networks for hidden Markov probability models. Technical Report MSR-TR-96-03, Microsoft Research, Redmond, WA.
Spiegelhalter, D.J., Lauritzen, S.L., 1990. Sequential updating of conditional probabilities on directed graphical structures. Networks 20, 579–605.
Varga, A.P., Moore, R.K., 1990. Hidden markov model decomposition of speech and noise. In: ICASSP-90, pp. 845–848.
Zhan, P., Waibel, A., 1997. Vocal tract length normalization for large vocabulary continuous speech recognition. Technical Report CMU-CS-97-148, School of Computer Science, Carnegie Mellon University.
Zweig, G., 1998. Speech recognition with dynamic Bayesian networks. Computer Science Division, University of California at Berkeley Dissertation.
Zweig, G., Russell, S., 1998. Speech recognition with dynamic Bayesian networks. In: AAA1-98.