DEPENDENCY MODELING WITH BAYESIAN NETWORKS IN A VOICEMAIL TRANSCRIPTION SYSTEM

Geoffrey Zweig and Mukund Padmanabhan
IBM T. J. Watson Research Center
{gzweig,mukund}@watson.ibm.com

ABSTRACT

In this paper we apply Bayesian networks to the problem of voicemail transcription. We use a Bayesian network system to test a variety of probabilistic models that model acoustic context in addition to phonetic state and acoustic observations. We use a context variable that has the ability to model contextual phenomena that are not implied by the linguistic sequence of phones (e.g. noise level or speech rate). In rescoring experiments, we are able to get a slight gain over a more standard system with a similar number of parameters. We obtained the best performance by conditioning the mixture coefficients on context, thus implementing a state-wise tied mixture system. In an utterance-clustering system, analysis of the learned parameters indicates that the context variable is highly correlated with $C0$ and $C1$.

1. INTRODUCTION

The Bayesian network formalism combines a graphical way of representing probabilistic models with a set of very general algorithms for computing with these models [6]. Bayesian networks generalize HMMs in allowing arbitrary sets of variables - with arbitrary conditioning relationships - to be associated with each time frame, while at the same time supporting functions analogous to those which render HMMs useful, particularly EM and Viterbi decoding. A key advantage of Bayesian networks is that the algorithms are general, so it is not necessary to derive new formulae or write new code for each new special case. Examples of ideas found in the literature which have a straightforward representation as Bayesian networks include probabilistic state classification [4], discrete HMM mixtures [7], and parallel model combination [1]. When the model under study has a highly factored state representation, i.e. associates several variables with each frame and conditions each on only a small subset of the others, Bayesian network algorithms also take advantage of these conditional independence relations to speed up the computations; [2] discusses cases where exponential speedups are possible.

Previous work [10, 9, 8] applied a Bayesian network system to large vocabulary isolated word recognition with significant improvements in accuracy. That work, however, used discrete output distributions and focused on isolated words. In this paper, we describe a Bayesian network system with continuous output distributions, and apply it to rescoring N-best lists generated by a LVCSR system. We find that it is possible to get small but not statistically significant improvements in accuracy, and that when used for utterance clustering, the system primarily distinguishes between high and low values of $C0$.

This work was supported by DARPA Grant MDA972-97-C-0012.

2. BAYESIAN NETWORKS

2.1. Definition

A Bayesian network is a directed acyclic graph that specifies a probability distribution over a set of random variables [6]. Each node in the graph corresponds to a variable, and the arcs that are incident on a node indicate the variables on which it is conditioned. We refer to node X_i's predecessors in the graph as its parents, and to the values of these variables as $Parents(X_i)$. The joint probability of a set of variable values is factored as:

$$P(X_1, X_2, \ldots, X_n) = \prod_i P(X_i|Parents(X_i)).$$

Associated with each node in the graph is a conditional probability function that returns the value of $P(X_i|Parents(X_i))$ for all possible values of X_i and its parents. In a fully discrete system, this can be done with simple table-lookup. In a system where X_i is a real valued feature vector, a mixture of Gaussians can be used. The last important concept is the distinction between an observation variable - whose value is given - and a hidden variable - whose value is unknown. The topologies required to do speech recognition with Bayesian networks have been treated in [8, 10, 9].

2.2. Algorithms

In this section we present a thumbnail sketch of the algorithms for computing the probability of a set of observations, the likeliest values for the hidden variables (Viterbi decoding), marginal distributions for the hidden variables, and EM. We present them in terms of a fully discrete system, and then state the changes that are necessary to accommodate continuous observation variables. The algorithms are defined in terms of a tree-structured network. General networks are handled in a two-step process that converts a non-tree structured network into a tree-structured one that represents the same probability distribution.

We refer to the observed variable values as evidence, and consider a three-way partitioning of the evidence with

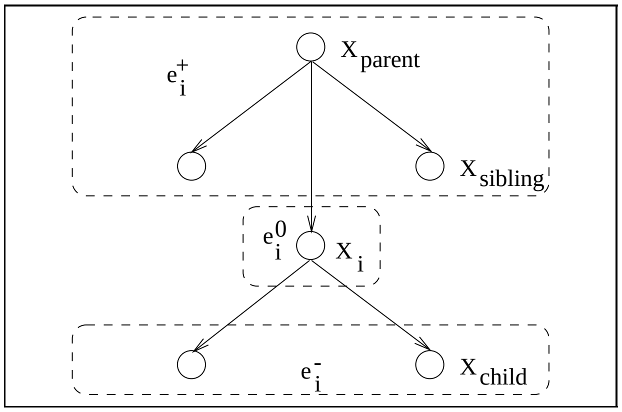

Figure 1: A tree of variables. The partitioning of the evidence is shown for X_i.

respect to each variable X_i. The evidence pertaining to X_i itself is referred to as e_i^0, and is the value of X_i when known, and the empty-set when X_i is hidden. The set of observed values for the variables in the subtrees rooted in X_i is e_i^-, and the remainder of the evidence is e_i^+ (see Figure 1). Because the variables are connected in a tree structure, the probability of all the evidence, and variable X_i taking value j, is:

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

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 any variable X_i,

- $P(Observations) = \sum_j \lambda_j^i * \pi_j^i$.
- $P(X_i = j | Observations) = \frac{\lambda_j^i * \pi_j^i}{\sum_j \lambda_j^i * \pi_j^i}$.

The recursions for computing the λs and πs are generalizations of the α and β procedures for HMMs, and sum over all the hidden-variable assignments that are consistent with the evidence, in the same way that the standard recursions sum over all possible paths through an HMM. The sums are expressed in terms of $CON(e_i^0)$, which refers to the values for X_i that are consistent with e_i^0. When X_i is a single variable, $CON(e_i^0)$ either contains all its possible values (when hidden) or just one (when observed). In section 2.2.1, we consider the more general case of composite variables whose values range over the Cartesian product of a set of constituent variables. In this case, $CON(e_i^0)$ refers to the set of cross-product values that are consistent with all known constituent values. Figure 2 presents the recursions. Finding the likeliest assignment of values can be done by replacing the sums by maximizations, analogous to Viterbi decoding with an HMM.

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

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

Figure 2: Inference in a tree.

2.2.1. General Graphs

Inference in non-tree-structured graphs such as that in Figure 4 is done by using a change-of-variables to convert the graph into an equivalent tree-structured one. Each new variable represents a subset of the old variables, and ranges over the Cartesian-product of its constituents. The new network must meet the following requirements [8]:

1. Each original variable must be found along with its parents in at least one of the new variables, and is said to be "assigned" to one such variable.
2. The new variables must be connected in a tree-structure such that if an original variable is found in two of the new variables, it is also present in every new variable along the path connecting the two.

Figure 3 illustrates an example of a general graph, and a satisfactory tree of composite variables.

Conditional probabilities in the new representation are defined as follows. Suppose Y_i is the parent of Y_j in the new tree. Let $\mathcal{V}$ be the set of original variables assigned to Y_j. The conditional probability $P(Y_j = m | Y_i = n)$ (with m and n being cross-product values) is defined as:

- 0, if $Y_j = m$ and $Y_i = n$ imply inconsistent values for a shared original variable.
- $\prod_{V \in \mathcal{V}} P(V | Parents(V))$, with the values for V and $Parents(V)$ implied by m, if $\mathcal{V} \neq \emptyset$. (In this case, n is not used.)
- 1, otherwise.

If Y_j is the root, there is no conditioning on Y_i, and only the last two items are relevant.

The probability of the observations can be computed as in Figure 2, using Ys in place of Xs. It is also possible

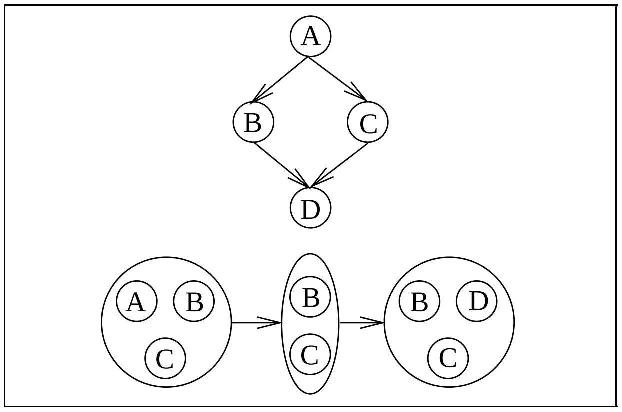

Figure 3: A simple non-tree structured Bayesian net (top), and a tree-structured equivalent (bottom). Nodes A, B, C are assigned to the left-most composite variable, and D to the right-most. In this case the tree has no side-branches.

to convert from the new representation back into the original: let Y_i be the new variable to which X_i is assigned. Let V^i_j be the set of Y's cross-product values corresponding to underlying variable assignments that include $X_i = j$. Then $P(X_i = j|Observations) = \sum_{w \in V^i_j} P(Y_i = w|Observations)$. Procedures for finding variable subsets and trees that satisfy the stated requirements can be found in, e.g., [6, 8].

2.2.2. EM

Let N_{ijk} be the number of times that variable X_i has value j and its parents are found in the kth possible configuration. In EM, θ_{ijk}, the probability that $X_i = j$ given that its parents have instantiation k, is estimated as $\frac{N_{ijk}}{\sum_j N_{ijk}}$ [3]. In order to calculate N_{ijk} we proceed as follows: Let Y_i be the new variable to which X_i is assigned. Let $\mathbf{V}^i_{jk}$ be the set of Y_i's cross-product values corresponding to underlying variable assignments that include $X_i = j$ and $Parents(X_i) = k$. Then

$$N_{ijk} = \sum_{w \in \mathbf{V}^i_{jk}} P(Y_i = w|Observations)$$

Estimating N_{ijk} from a collection of examples requires summing the individual estimates for each example.

2.2.3. Continuous Observation Variables

The special case of continuous variables is easy to deal with when these variables are always observed, have only discrete variables as parents, and are never themselves parents. It is analogous to the extension of discrete HMMs to continuous-valued feature vectors. In this case, a variable value j is a real-valued vector, and in λ^i_j and π^i_j it is taken to refer to the single observed (but continuous) value. Sums involving j reduce to a single term, and conditional probabilities of the form $P(X_i = j|X_p = v)$ can be represented with Gaussian mixtures. In EM the count N_{ijk} is

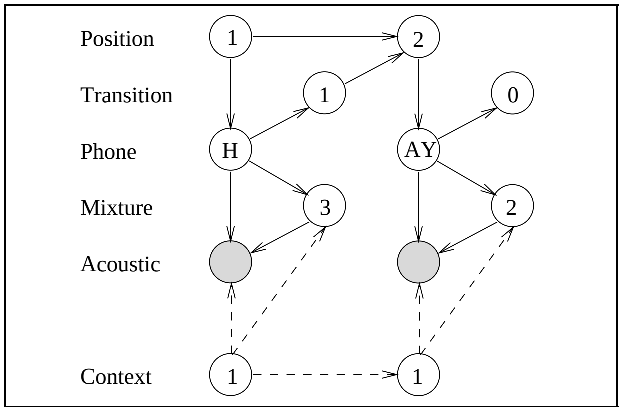

Figure 4: Two slices of the Bayesian network. Hidden variables are unshaded, but shown with representative assignments.

combined with the vector $\mathbf{j}$ to update the sum of first and second order statistics.

3. EXPERIMENTAL RESULTS

This section presents results for a Bayesian network system used to rescore test hypotheses generated by the standard IBM system. We present results for a development set of 43 phone-mail messages with 1986 words, and a test set with 86 messages and 6,925 words. We generated the 100 best hypotheses using a system with approximately 100k Gaussians and a context-sensitive phone alphabet of 2709 units. Acoustic processing consisted of computing the first 13 cepstral coefficients, their deltas, and double-deltas. The baseline system is described in more detail in [5]. For the test set, the error rate of the original system - calculated by always selecting the first hypothesis - is 43.67%. The best possible error rate - calculated by always selecting the hypothesis with fewest errors - is 40.75%.

We present results for five different network structures. The basic network structure is shown in Figure 4. The conditional probabilities of the position, transition, and phone variables have been described previously [10, 9, 8], and ensure that variable assignments correspond to valid segmentations of an utterance according to the phone sequence specified by the training or testing hypothesis. An observation vector is modeled with a single Gaussian. In order to create Gaussian-mixture output distributions, we condition the observation variable on an explicit mixture component variable.

The five variants we studied differ in which variables were conditioned on a binary context variable: 1) The observation variables; 2) Mixture component variables; 3) Both; 4) Neither (emulating an HMM); 5) Both, with the auxiliary variable constrained not to switch values in the course of an utterance, thus implementing utterance clustering.

If the same number of mixture components is used, conditioning the observation variable on the binary context variable will double the number of parameters. In order to

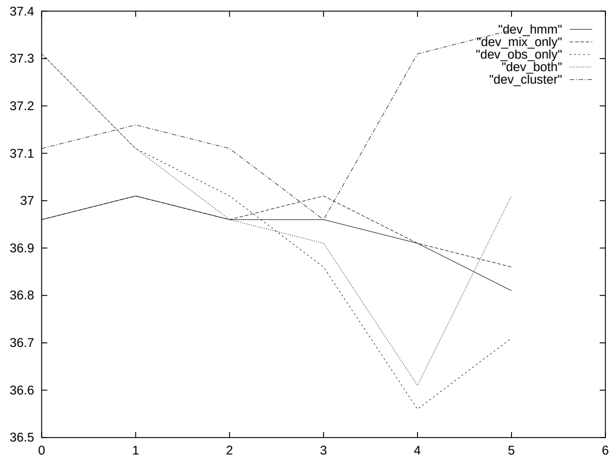

Figure 5: Learning curves that show the word error rate as a function of EM iteration number, on the development set.

HMM (4)	Obs (1)	Mix (2)	Both (3)	Cluster (5)	Do Nothing
43.41%	43.42	43.34	43.42	43.36	43.67

Table 1: Word error rates on the test set, after rescoring the top-100 hypotheses. Sytem number is shown on the second line. “Do Nothing” refers to selecting the first item on the N-best list.

ensure a fair comparison, systems 1, 3, and 5 had approximately half the number of mixture components. These systems had about 48k Gaussians, while systems 2 and 4 had 44k. Learning curves on the development set are presented in Figure 5. We see that most of the systems begin to overtrain by four iterations, with the exception of the HMM and mixture-only systems. Results on the test set using the optimal number of EM iterations (determined from the development set) are presented in Table 1. The tied-mixture system did best, though by a tiny margin.

Interestingly, the mixture-only system is in essence a tied-mixture system on the state level. In usual tied-mixture systems, all the states share the same pool of Gaussians, and differ only in the mixture coefficients. In this system, each state has its own distinct pool of Gaussians, but the value of the context variable selects one of two output distributions, by choosing between one of two sets of mixture coefficients.

In order to interpret the model parameters, we examined the two sets of Gaussians - one for each auxiliary variable value - and found a systematic pattern in the clustering network (system 5). In this case, one of the sets of Gaussians tended to have a systematically larger value for the $C0$ mean. This is shown in Figure 6. A similar pattern was found for $C1$.

4. CONCLUSION

We used a Bayesian network system to test several probabilistic models that differed in the way in which mixture component and observation variables were conditioned on an auxiliary context variable. Overtraining proved to be a problem in the systems in which the observations were directly conditioned on the context. Conditioning the mixture component variable alone resulted in a state-wise tied mixture system, and produced a small improvement.

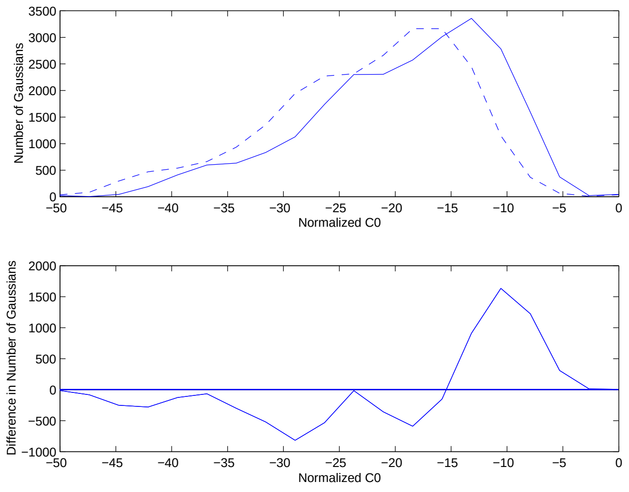

Figure 6: Top: Histograms of the $C0$ component of the Gaussian means when the context value is 0 (solid line) and 1 (dashed). Bottom: The difference. Curves are for the clustering network.

5. REFERENCES

[1] M.J.F. Gales and S. Young. An improved approach to the hidden markov model decomposition of speech and noise. In *ICASSP-92*, pages I–233–I–236, 1992.

[2] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29(2/3), 1997.

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

[4] Xiaoqiang Luo and Frederick Jelinek. Probabilistic classification of HMM states for lvcsr. In *Proc. ICASSP*, 1999.

[5] M. Padmanabhan, B. Ramabhadran, and S. Basu. Speech recognition performance on a new voicemail transcription task. In *ICSLP-98*, 1999.

[6] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[7] S. Tsakalidis, V. Digalakis, and L. Neumeyer. Efficient recognition using subvector quantization and discrete-mixture hmms. In *ICASSP-99*, 1999.

[8] Geoffrey Zweig. *Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, University of California, Berkeley, Berkeley, California, 1998.

[9] Geoffrey Zweig and Stuart Russell. Probabilistic modeling with Bayesian networks for automatic speech recognition. In *ICSLP-98*, 1998.

[10] Geoffrey Zweig and Stuart Russell. Speech recognition with dynamic Bayesian networks. In *AAAI-98*, 1998.