

# The Role of Boosting and Structure Learning in Dynamic Bayesian Networks for Multi-modal Speaker Detection

Submission Number 768

## Abstract

*We are interested speech-based interfaces for a Smart Kiosk, a free-standing computer system which provides information and services to multiple people in a public space. Bayesian networks are an attractive modeling tool for human sensing, as they combine an intuitive graphical representation with efficient algorithms for inference and learning. Earlier work demonstrated that boosted parameter learning could be used to improve the performance of Bayesian network classifiers for speaker detection.*

*In this paper we present a variation of AdaBoost based on max classifier selection that improves the performance of Bayesian network classifiers. We introduce a new boosting algorithm that incorporates structure learning. We compare its classification accuracy with that of classical structure learning and parameter boosting on fixed network structures. We present experimental results for speaker detection as well as experiments with the “chess” dataset from the UCI repository.*

## 1 Introduction

Human-centered user-interfaces based on vision and speech present challenging sensing problems in which multiple sources of information must be combined to infer the user’s actions and intentions. Statistical modeling techniques therefore play a critical role in system design. Dynamic Bayesian network (DBN) models are an attractive choice, as they combine an intuitive graphical representation with efficient algorithms for inference and learning. DBNs are a class of graphical probabilistic models which encode dependencies among sets of random variables evolving in time. Examples of DBNs include Kalman filters and HMMs. Previous work has demonstrated the power of these models in fusing video and audio cues with contextual information and expert knowledge [15, 13, 3].

Speaker detection is a particularly interesting example of a multi-modal sensing task which can serve as a test-bed for DBN research. In an open mike speech-based interface, one needs to identify the speaker and discriminate speech directed to the interface from

conversations with other users and background noise. Both audio- and video-based sensing can provide useful cues [6, 7]. Figure 4 gives an example of a DBN model for speaker detection. We are interested in network models that combine “off-the-shelf” vision and speech sensing with contextual cues such as the state of the interaction. Previous work has demonstrated promising results for speaker detection using the Bayesian network approach [15, 17].

The challenge in applying DBN models to classification tasks like speaker detection is to develop effective discriminative learning algorithms. Classical parameter learning algorithms for DBN’s are unsupervised, in the sense that all nodes in the network are treated equally. However, when DBN’s are used as classifiers we would prefer a supervised approach, in which the classification node is identified and parameter learning can be optimized for classification performance. For example, in Figure 2, the S node models the classifier state “speaking to the interface”.

In previous work, we used the AdaBoost algorithm [19] to develop a DBN parameter learner that was tuned for classification accuracy. In the speaker detection example, boosting improved the DBN classifier by 15%.

In this paper we significantly extend these previous results in two ways. First, we expand the learning task to include structure learning. Given the nodes in the DBN model, we search over the set of possible graph structures. This allows us to compensate for possible biases or inaccuracies in the hand-specified graph of Figure 4. We present a novel algorithm for boosted structure learning which extends our previous results on boosted parameter learning. Second, we describe a mixture of experts interpretation of the output of AdaBoost, which leads to a max-based classifier selection approach. We test these new algorithms on the speaker detection task and the chess data set from the UCI repository [2].

## 2 Speaker Detection

Detecting when users are speaking is an important component of an open mike speech-based user-interface. The challenge is to identify a speaker’s focus of attention, and discriminate speech directed to the interface from

conversations with other users and background noise. Both audio- and video-based sensing can provide useful cues.

We assume that a speaker will be facing the kiosk, moving their lips, and producing speech. Visual cues can be useful in deciding whether the person is facing the kiosk and whether they are moving their lips. However, they are not capable on their own to distinguish a speaker from an active listener, who may be facing the kiosk while smiling or nodding. Audio cues can detect the production of speech. However, simple audio cues can not distinguish speech directed to the kiosk from speech



Figure 1 The Smart Kiosk

directed at one's neighbors. In addition, contextual information describing the state of the interface also has bearing on speaker detection. For instance, in certain contexts the user may not be expected to speak at all.

The Smart Kiosk [16, 4] provides an interface which allows the user to interact with the system using spoken commands. The public, multi-user nature of the kiosk application domain makes it ideal as an experimental setup for speaker detection task. The kiosk (see Figure 1) has a camera mounted on the top that provides visual feedback. A microphone is used to acquire speech input from the user.

We have analyzed the problem of speaker detection in a specific scenario of the Genie Casino Kiosk. This version of kiosk simulates a multiplayer blackjack game (see Figure 8 for a screen capture.) The user uses a set of

spoken commands to interact with the dealer (kiosk) and play the game.

Audio and visual information can be obtained directly from the two kiosk sensors. We use a set of five “off-the-shelf” visual and audio sensors: the CMU face detector [18], a Gaussian skin color detector [21], a face texture detector, a mouth motion detector, and an audio silence detector. A detailed description of these detectors can be found in [17].

### 3 Dynamic Bayesian Networks for Speaker Detection

A Bayesian network (BN) is a graphical representation of a factored joint probability distribution for a set of random variables. Figure 2 gives an example of a BN for the speaker detection problem. Each node is a variable. The *speaker* node, for example, equals one whenever a user is speaking to the kiosk and zero otherwise. It influences three other hidden nodes, which describe whether the speaker's face is *visible* and *frontal*, and whether *speech* is being produced.

The six measurement nodes, drawn with dashed lines in Figure 2, encode video, audio, and contextual cues. The *context* node measures the current state of the user's interaction with the kiosk. The other five measurement nodes represent the output of specific video and audio processing modules. *Face detector*, for example, is the binary output of the CMU Face Detector [18]. It equals one whenever the video input contains a frontal, upright face.

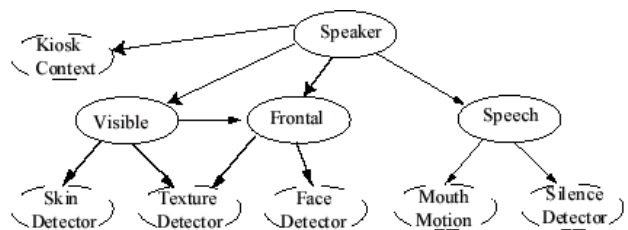
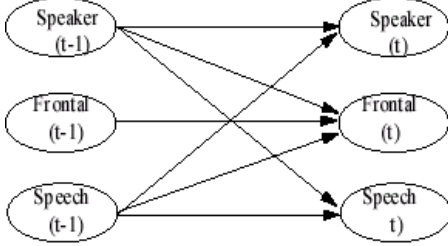


Figure 2 Static network for speaker detection

The arcs between the nodes are parameterized by conditional probability distributions that model dependencies between variables. The arc between the two binary variables *speaker* and *visible*, for example, stores the two-by-two conditional probability table (CPT),  $P(\text{visible}/\text{speaker})$ . We let  $B_\theta$  denote the total set of CPT parameters. Given the CPT's and a set of measurements, efficient inference algorithms can be used to compute the distribution over the hidden nodes given the

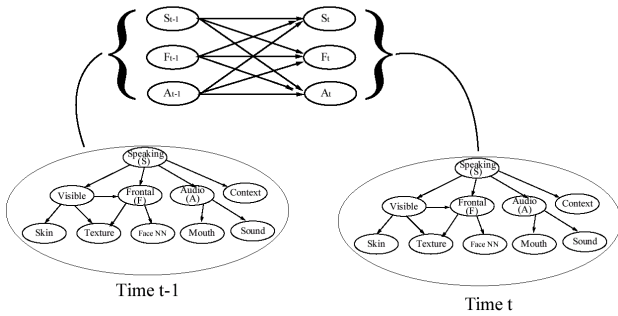
measurements [14]. The speaker detector output is given by a likelihood ratio test on the *speaker* variable. See [17] for a more detailed description.

Adding temporal dependencies between variables in a



**Figure 3** Temporal dependencies between the speaker, speech and frontal nodes

BN results in a dynamic Bayesian network (DBN). Figure 3 gives an example for speaker detection. Each arc denotes a dependency between variables in two “slices” of the network at consecutive times. For example, the top-most arc represents  $P(\text{speaker}_t/\text{speaker}_{t-1})$ . The probability distribution is defined by the parameters of the Markov



**Figure 4** Two time slices of the dynamic Bayesian network for speaker detection

model: the matrix  $A$  of transition probabilities and the initial state distribution  $\pi$ .

Combining the temporal dependencies in Figure 3 with the static BN in Figure 2 gives the complete DBN for speaker detection, illustrated in Figure 4. We can represent the network as a tuple  $(B_s, \theta)$ , where  $B_s$  encodes the *structure* (i.e. the topology) of the network and  $\theta = \{B_\theta, A, \pi\}$  is the set of parameters. In these examples,  $B_s$  is specified manually.

The parameters can be learned from a training data set  $D$  by computing

$$\theta^* = \arg \max_{\theta} P(D/\theta)P(\theta), \quad (1)$$

where  $P(\theta)$  is a prior distribution. When all of the nodes are observed, this computation can be done in closed-form [12].

Learning is particularly simple for the network of Figure 4. Let  $z$  denote the four hidden states and  $y$  denote the six measurements. Let  $Z_T = \{z_1, z_2, \dots, z_T\}$  be the sequence of  $T$  hidden states and  $X_T$  the corresponding sequence of measurements. Then we have

$$P(Z_T, Y_T, \theta) = P(Y_T | Z_T, B_\theta)P(Z_T | A, \pi).$$

Thus the parameters  $B_\theta$  can be determined by counting how often particular combinations of hidden state and measurement values occur. In this simplest case the parameters are simply the counts in a histogram of the training data. We can further expand the second term:

$$P(Z_T | A, \pi) = \prod_i P(z_i | z_{i-1}, A, \pi).$$

Thus the transition matrix  $A$  can be viewed as a second histogram which counts the number of transitions between the hidden state values over time.

Inference is equally straight-forward using the standard forward-backward algorithm. See [15] for details.

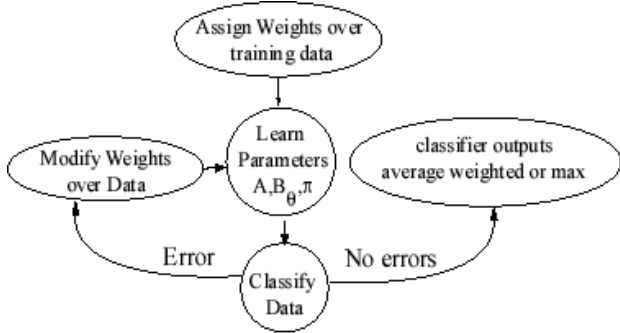
## 4. Bayesian Network Classifiers

DBN models are an appealing framework for complex inference problems because they are interpretable, composable, and generative. Posthoc analysis of learned parameters and network structure is an important source of insight into network performance. Such insight can be difficult to obtain in directly supervised learning approaches such as neural networks. Second, it is fairly easy to compose large Bayesian network models by combining subgraphs. This makes it possible to reuse and extend modeling components without retraining an entire model for each new problem [8]. Third, because the Bayesian network models a joint probability distribution, sampling can be used to generate synthetic data. This is another source of insight into network performance.

However, the straightforward approach of learning BN models from training data and then applying them to a classification task can result in poor performance [11]. To understand why, consider a dataset of  $N$  records  $D = \{x_1, \dots, x_N\}$ . Let  $x_i = \{s_i, y_i\}$ , where  $s_i$  is the classification node (eg. the speaker node in Figure 2) and  $y_i$  is the observations for record  $i$ . Substituting into Equation 1 we have

$$\begin{aligned} \theta^* &= \arg \max_{\theta} \prod_i P(s_i, y_i | \theta)P(\theta) \\ &= \arg \max_{\theta} \sum_i \log P(s_i | y_i, \theta) + \log P(y_i | \theta) + \log P(\theta) \end{aligned} \quad (2)$$

The classification performance of the network is governed by the first term in Equation 2, known as the conditional log likelihood. Since the parameter estimate maximizes the total posterior, it is not guaranteed to give



**Figure 5** Parameter boosting for DBNs

an optimal estimate for the conditional likelihood under the structure  $B_s$ . If the structure is incorrect, the resulting classifier may not generalize well during testing. Unfortunately, it does not seem to be possible to extend the closed form solutions for Equation 2 to the conditional likelihood term in isolation. See [11] for details.

One solution to this problem is to use boosting to improve the classification performance. In the Adaboost algorithm of Schapire et al. [14], performance is improved by linearly combining a sequence of weak classifiers, each of which is trained to correct the mistakes of the previous one on the training data. In previous work [15], we used this approach to improve the speaker detection performance of the DBN classifier of Figure 4.

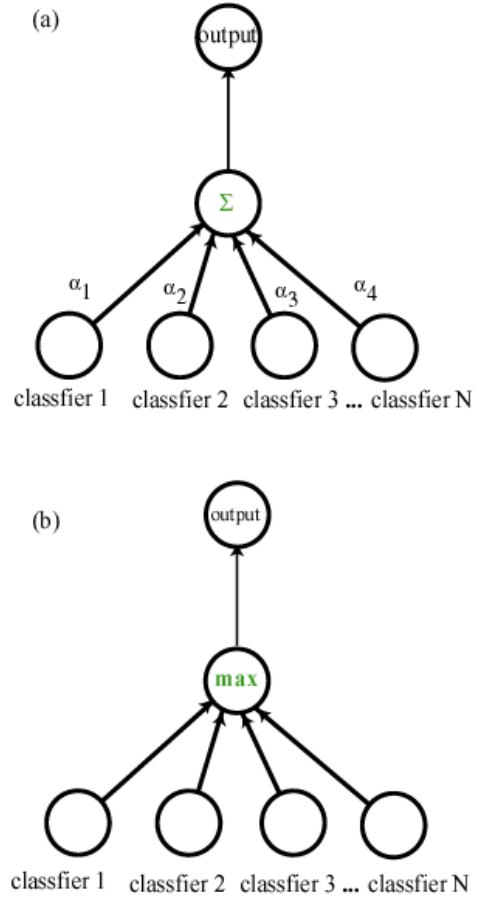
The boosting algorithm for DBN parameter learning is illustrated in Figure 5. The key point is that boosting modifies the classifier design by changing the weights on the training data according to the classifier’s performance. This is attractive as it means we can retain the efficient parameter learning algorithms for DBNs.

Boosting has a particularly simple interpretation for network of Figure 4. For simplicity, consider just the classifier node  $s$  (i.e. speaker) and the measurements  $y$ . Boosting modifies  $B_\theta$  according to the distribution  $P(y/s, D_k)$  where  $D_k$  is the reweighted training data at iteration  $k$  of boosting. Intuitively, boosting will increase or decrease the weighted counts in a particular bin ( $s=s^i, y=y^j$ ) of the histogram depending on whether the classification given by  $P(s=s^i | y=y^j)$  is incorrect or correct.

Similarly, boosting modifies  $A$  according to  $P(s_i / s_{-i}, D_k)$ . Intuitively, boosting will increase or decrease the

weighted counts for a pair of state transitions ( $A_{ij}, A_{jk}$ ) based on the classification performance for  $s_i=e^j$ . This can be viewed as an error-driven duration density model for the Markov chain, and it seems to be the primary source of performance improvement in the classifier of Figure 4.

Now consider the decision boundary between ( $s=0, y=y^j$ ) and ( $s=1, y=y^j$ ). Since all of the variables are discrete, boosting can only effect the decision by changing the sign at the boundary. Given some initial



**Figure 6** Combining classifiers (a) weighted average (b) max selection

distribution of counts between the two bins, boosting will tend to drive the distribution towards  $(0.5, 0.5)$ . Until this threshold is reached, boosting will not change the decision boundary.

As a consequence, the final classifier produced by averaging the reweighted classifiers may not give significant improvement. The combination scheme used in AdaBoost is illustrated in Figure 6(a). An alternative combination scheme which we explored in this paper is

illustrated in Figure 6(b). Rather than taking the average, we select the output of the classifier which maximizes the likelihood of the test probe. We can view this as a mixture of experts.

## 5. Automatic Learning of Network Structure

The network structures in Figures 2, 3, and 4 were manually specified using knowledge about the problem and sensors. However, we may introduce unwanted bias while designing the network topology. Also, the task becomes harder when we start modeling more complicated networks with many more features. One solution to the problem would be to learn the network structure automatically from the data.

Structure learning for both static and dynamic Bayesian networks has been studied by a number of researchers [5, 9, 10,12]. The main objective of a structure learning algorithm is to find the structure that is best supported by the data. In order to find the best structure the algorithm must compare network structures and also search over structures in a computationally efficient way. If  $D$  is the dataset and  $B_{s_1}$  and  $B_{s_2}$  are two network structures. By computing the posterior probabilities for the network structures and taking their ratio  $P(B_{s_1}|D) / P(B_{s_2}|D)$  we can rank the candidate structures. We can use the following equivalence relationship to rank the structures:

$$\frac{P(B_{s_1} | D)}{P(B_{s_2} | D)} = \frac{\frac{P(B_{s_1}, D)}{P(D)}}{\frac{P(B_{s_2}, D)}{P(D)}} = \frac{P(B_{s_1}, D)}{P(B_{s_2}, D)} = \frac{P(D | B_{s_1})P(B_{s_1})}{P(D | B_{s_2})P(B_{s_2})}$$

If the prior over the structure is uniform then maximum likelihood solution will be the structure that maximizes the following quantity –  $P(D|B_{s_i})$ . Since the number of possible network structure is doubly exponential in the number of nodes an exhaustive search over all structures is not possible. Therefore, some heuristic method is needed to maximize  $P(D, B_{s_i})$ . For example a node ordering heuristic is used in K2 - a structure learning algorithm proposed by Cooper and Herskovits [5]. K2 is a greedy search algorithm that tries to find the structure that maximizes the data likelihood given the structure. This algorithm requires the knowledge of the node orderings of the network and the maximum number of parents a node can have. The algorithm restricts search for parents of a given node by its node number and the number of parents it is allowed to have. One way to overcome the node ordering constraint is to do a MCMC

search over node orderings and pick the node order that gives in the best structure in terms of data likelihood [22].

## 6. Boosted Structure Learning

Analogous to the parameter learning case, a standard structure learning algorithm such as K2 is not guaranteed to produce an optimal classifier. In this section we describe a new algorithm for boosting for both the network structure and network parameters.

Table 1 outlines our algorithm for boosted structure

<p>Given : <math>D\{(s_i; y_1), \dots, (s_T; y_T)\}</math>; where <math>y_i</math> is an observation vector and <math>s_i</math> is the corresponding label of the hidden state</p> <p>Initialize <math>P_D^1(i) = 1/T</math>;</p> <p>For <math>k = 1, \dots, K</math></p> <ul style="list-style-type: none"> <li>• Use <math>P_D^k</math> as the weight over the training samples.</li> <li>• Learn the static Bayes net <math>B_{S_{k_i}}</math> for the weighted training samples <ul style="list-style-type: none"> <li>If doing parameter boosting only use the pre-specified structure</li> </ul> </li> <li>• Train the static BN using the learnt <math>B_{S_k}</math> or to obtain parameters <math>B_{\theta_k}</math>.</li> </ul> <p>If using static BN</p> <ul style="list-style-type: none"> <li>• Use the learned BN (<math>B_{S_k}, B_{\theta_k}</math>) to decode the hidden state sequence <math>(s_1, \dots, s_T)</math> given <math>(y_1, \dots, y_T)</math> as the input <math display="block">\hat{s}_i = \arg \max_i P(s_i = i   y_i)</math> </li> </ul> <p>If using dynamic BN</p> <ul style="list-style-type: none"> <li>• Use the DBN learning algorithm to obtain the transition probability matrix <math>A</math> for fixed <math>B_{S_k}</math> and <math>B_{\theta_k}</math>.</li> <li>• Use the learned DBN <math>\theta = (A, B_{\theta_k}, \pi)</math> to decode the hidden state sequence <math>(s_1, \dots, s_T)</math> given <math>(y_1, \dots, y_T)</math> as the input <math display="block">\hat{s}_i = \arg \max_i P(s_i = i   y_1, \dots, y_T)</math> </li> <li>• Choose <math>\alpha_k = 0.5 * (\frac{1 - \epsilon_k}{\epsilon_k})</math>, where <math>\epsilon_k = \sum_i P(\hat{s}_i \neq s_i)</math></li> <li>• Update : <math display="block">\text{if } \hat{s}_i \neq s_i \text{ then } P_D^{k+1}(i) = \frac{P_D^k(i) \exp(\alpha_k)}{Z_k}</math> <math display="block">\text{if } \hat{s}_i = s_i \text{ then } P_D^{k+1}(i) = \frac{P_D^k(i) \exp(-\alpha_k)}{Z_k}</math> <p>where <math>Z_k</math> is the normalization factor</p> </li> </ul> <p>The final output can be a weighted combination of the classifier outputs (AdaBoost) or equal to output of the classifier with the highest output probability</p>
---

**Table 1** Boosting algorithm for structure and parameter boosting

learning and Figure 7 gives the flow diagram. If we were to only boost the Bayes net parameters assuming known

structure we would skip structure learning step (see Figure 5). In our algorithm we only learn the structure for

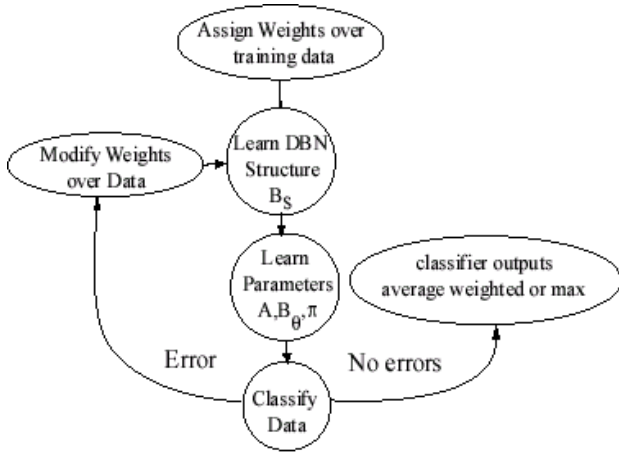


Figure 7 Structure and Parameter boosting for DBNs

the static part of the network and the temporal structure is fully connected as specified earlier. The final classifier can be generated by weighted averaging or max-selection, as described in Section 4.

## 7. Experiments and Results

We have conducted experiments on two separate datasets – (i) the speaker dataset and (ii) the “chess” dataset available from the UCI machine learning repository [2]. The speaker dataset is comprised of five sequences of a user playing the blackjack game in the Genie Casino

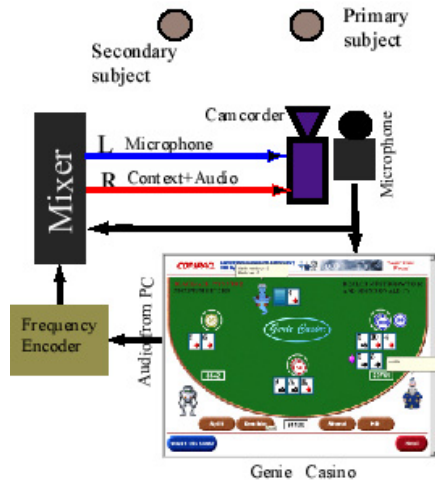


Figure 8 Data collection setup for Genie Casino kiosk

Kiosk setup. The experimental setup is depicted in Figure 8. The sequences are of varying duration (from 2000-3000 samples) totaling to approximately 10000 frames. The chess has no temporal component to it and was mainly used to evaluate the performance of our algorithm against a standard dataset. The dataset has 36 attributes or features values and two output classes.

For the speaker dataset each sequence included audio and video tracks recorder through a camcorder along with frequency encoded contextual information. The visual and audio sensors were then applied to audio and video streams. Because some of the sensors provide continuous estimates of their respective functions, decision thresholds were determined for each sensor that yield binary sensor states (e.g., silence vs. no silence.) These discretized states were then used as input for the DBN model.

The experimental results for both datasets are summarized in Table 2. Our first results examine the performance of standard structure learning on the speaker detection task. Figure 9 and 10 show two example of the learned network structures using K2 for the speaker dataset. We observe in the first structure that the nodes *visible* and *skin* are disconnected from the rest of the network. Because of the way the data was collected these

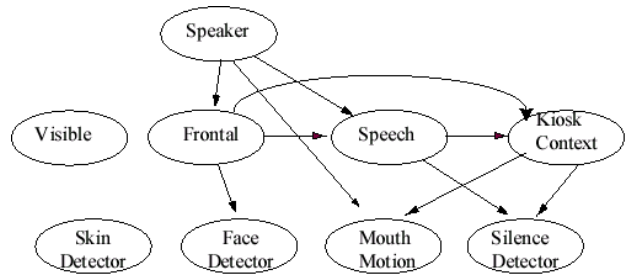


Figure 9 Learned Structure using original node ordering

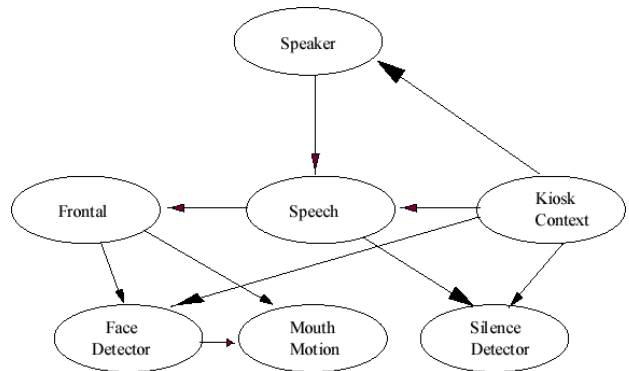


Figure 10 Learned structure using node order from MCMC search

to nodes were always present, thus did not provide any useful information to our classifier. The first structure used the same node ordering as the hand-coded structure. The second network structure was learnt using a MCMC search over node orderings and picking the node order that resulted in the best structure in terms of data likelihood. From figure 10 we see that the node connections make sense – *face detector* and *mouth motion* influence the *frontal face* node, *speech* node is influenced by the *silence detector* node etc.

	Speaker Dataset	Chess Dataset
Static BN Known structure No boosting	69%	88%
Static BN Known structure AdaBoost	69%(0%)	88%(0%)
Static BN Known structure Max selection	71%(+2%)	92%(+4%)
Static BN Learned structure No boosting	87%(+16%)	94%(+6%)
Static BN Learned structure AdaBoost	87%(+16%)	94%(+6%)
Static BN Learned structure Max selection	88%(+17%)	96%(+8%)
DBN Learned structure No boosting	90%(+19%)	---
D BN Learned structure AdaBoost	91%(+20%)	---
D BN Learned structure Max selection	92%(+21%)	---

**Table 2** Experimental results for the speaker dataset and chess dataset

During testing only the sensor outputs were presented and inference was done to obtain the values for the hidden nodes. Mismatch in any of the three hidden nodes (speaker, frontal, audio) is considered to be an error. Cross validation was done by choosing different training and test data.

For both the speaker and “chess” datasets, we measured classification accuracy for (i) static Bayesian network without boosting (ii) static Bayesian network with AdaBoost (iii) static Bayesian network with max-selection boosting. For the speaker dataset we also

measure the classification accuracy for (iv) dynamic Bayesian Network without boosting (v) dynamic Bayesian network with AdaBoost and (vi) dynamic Bayesian network with max-selection boosting. Table 2. shows the performance results for both datasets. The number inside the parenthesis shows the improvement in classification accuracy from our baseline model - the static Bayesian network.

## Conclusion

We extend the AdaBoost algorithm for dynamic Bayesian networks to include structure learning which produces significant performance improvement over both classical structure learning and boosted parameter learning. We believe this is the first use of boosting in structure learning. We also give some new insights into the performance of parameter boosting. Our algorithms have been validated on a real world speaker detection dataset and the standard “chess” dataset from the UCI repository.

## References

- [1] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Estimating hidden Markov model parameters so as to maximize speech recognition accuracy," *IEEE Transactions on Speech & Audio Processing*, vol. 1, pp. 77-83, 1993.
- [2] C.L. Blake, and C.J. Merz, (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mllearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
- [3] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," *Proceedings of Computer Vision and Pattern Recognition*, pp. 994-9, 1997.
- [4] A. Christian, and B. Avery, "Digital smart kiosk project," *ACM SICHI*, vol. 98, pp. 155-62, 1998.
- [5] G. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, pp. 309-347, 1992.
- [6] R. Cutler and L. Davis, "Look who's talking: speaker detection using video and audio correlation," presented at *IEEE International Conference on Multimedia Expo (ICME)*, New York, 2000.
- [7] Darrell, T., Fisher, J., Viola, P., Freeman, B., Audio-visual Segmentation and the Cocktail Party Effect, *Proceedings of the International Conference on Multimodal Interfaces*, Beijing, October 2000.
- [8] T. Dean, and K. Kanazawa, "A model for reasoning about persistence and causation," *Computational Intelligence*, vol. 5, pp. 142-50, 1989.

- [9] N. Friedman, "Learning belief networks in the presence of missing values and hidden variables," presented at *International Conference in Machine Learning*, 1997.
- [10] N. Friedman, K. Murphy, and S. Rusell, "Learning the Structure of Dynamic Probabilistic Networks," presented at *Uncertainty in Artificial Intelligence*, Madison, Wisconsin, 1998.
- [11] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, pp. 2-3, 1997.
- [12] D. Heckerman, "A tutorial on learning with Bayesian networks," Microsoft Research MSR-TR-95-06, 1995.
- [13] S. Intille, and A. Bobick, "A framework for recognizing multi-agent action from visual evidence," *Proceedings Sixteenth National Conference on Artificial Intelligence*, vol. 99, pp. 518-25, 1999.
- [14] F. V. Jensen, *An Introduction to Bayesian Networks*. New York: Springer-Verlag, 1997.
- [15] Anonymous authors, "Multimodal speaker detection using error feedback dynamic Bayesian networks," 2000.
- [16] Anonymous authors, "Vision for a smart kiosk," 1997.
- [17] Anonymous authors, "Vision-based speaker detection using Bayesian networks," 1999.
- [18] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pp. 203-8, 1996.
- [19] R. Schapire, and Singer Y, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, pp. 297-336, 1999.
- [20] H. Schwenk, "Using boosting to improve a hybrid HMM/neural network speech recognizer," *Proceedings ICASSP*, pp. 1009-12, 1999.
- [21] J. Yang and A. Waibel, "A real-time face tracker," *Proceedings 3rd Workshop on Appl. of Comp. Vision*, 142-147, Sarasota, FL, 1996.
- [22] N. Friedman and D. Koller, "Being Bayesian about network structure", *Proceedings 16<sup>th</sup> Conference on Uncertainty in AI (UAI)*, 2000.