Appears in *Proc. of Intl. Conf. on Pattern Recognition*, pages III:789-794, Quebec City, Canada, Aug. 11-15, 2002.

Boosting and Structure Learning in Dynamic Bayesian Networks for Audio-Visual Speaker Detection

Tanzeem Choudhury[1], James M. Rehg[2], Vladimir Pavlović[3], and Alex Pentland[1]

[1]Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{tanzeem,sandy}@media.mit.edu

[2]College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
rehg@cc.gatech.edu

[3]College of Engineering
Boston University
Boston, MA 02215
vladimir@bu.edu

Abstract

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 has demonstrated that boosted parameter learning could be used to improve the performance of Bayesian network classifiers for complex multi-modal inference problems such as speaker detection. In speaker detection, the goal is to use video and audio cues to infer when a person is speaking to a user interface. In this paper we introduce a new boosted structure learning algorithm based on AdaBoost. Given labeled data, our algorithm modifies both the network structure and parameters so as to improve classification accuracy. We compare its performance to both standard structure learning and boosted parameter learning on a fixed structure. We present results for speaker detection and for the UCI "chess" dataset.

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. Dynamic Bayesian network (DBN) models are an attractive modeling 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 [14, 11, 2].

Speaker detection is a particularly interesting example of a multi-modal sensing task which can serve as a test-bed for DBN research [16, 9]. 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 [4, 5].

The challenge in applying DBN models to 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 learning is optimized for classification performance.

In previous work, the AdaBoost algorithm [18] was used to develop a DBN parameter learner that was tuned for classification accuracy [14]. In the speaker detection example, boosting improved the performance of 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 hand-specified models. We present a novel algorithm for boosted structure learning which extends our previous results on boosted parameter learning. Second, we describe a variation of AdaBoost which uses a max-based classifier selection approach to determine the output. We test these new algorithms on the speaker detection task and the chess data set from the UCI repository [1].

2. DBN for Speaker Detection

The context for our work the development of an open-mike speech interface for a Smart Kiosk [15]. The kiosk has microphone and camera inputs and a graphical avatar for output. 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 kisok 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 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.

Figure 1 gives an example of a Bayesian network for the speaker detection problem. Each node is a variable. The *hidden* speaker node, for example, equals 1 whenever a user is speaking to the kiosk and -1 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 gray lines in Figure 1, 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 [17]. Efficient inference algorithms can be used to compute a distribution over the speaker node given the measurements [12]. The speaker detector output is given by a likelihood ratio test on the speaker variable (see [16] for details).

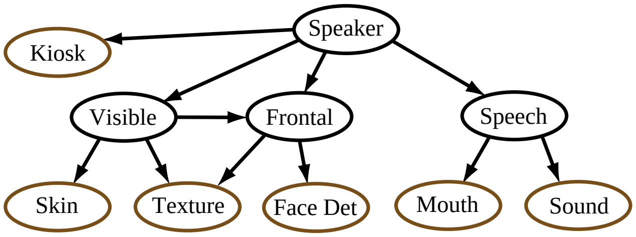

Figure 1. Static Bayesian network for speaker detection.

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_ϕ denote the total set of CPT parameters. Adding temporal dependencies between variables in a BN results in a dynamic Bayesian network (DBN). Figure 2 gives an example for speaker detection. Additional arcs have been placed between the three hidden nodes *speaker*, *frontal*, and *speech*. Each arc denotes a dependency between variables in two "slices" of the network at consecutive times. The probability distribution is defined by the parameters of a Markov model: a matrix A of transition probabilities and an initial state distribution π.

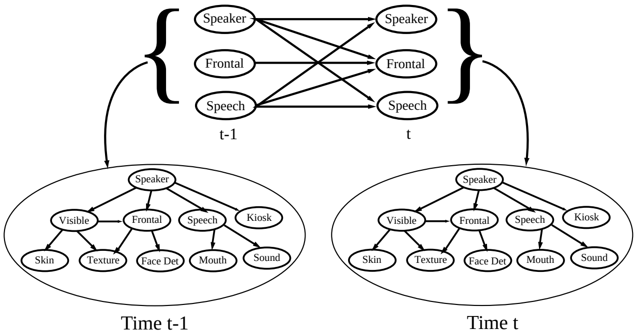

Figure 2. Dynamic Bayesian network for speaker detection.

We can describe the DBN model of Figure 2 as the tuple (B_s, θ), where B_s encodes the structure (i.e. the topology) of the network and $\theta = \{B_\phi, A, \pi\}$ is the set of network parameters. We can decompose B_s into $\{B_c, B_d\}$, where B_c is the structure of the *static network* (in Figure 1) and B_d specifies the *temporal arcs* between time slices (in Figure 2). In this instance, B_s has been specified manually. The parameters can be learned from a training data set D by computing

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

where $P(\theta)$ is a prior. When all of the nodes are observed, this computation can be done in closed-form [10].

Learning is particularly simple for the network of Figure 2. Let z denote the four hidden states and y denote the six measurements. Let $Z_T = \{z_1, z_2, ..., z_T\}$ be the sequence of T hidden states and Y_T the corresponding sequence of measurements. Then we have:

$$P(Z_T, Y_T, \theta) = P(Y_T|Z_T, B_\phi)P(Z_T|A, \pi) \tag{2}$$

Thus the parameters B_ϕ 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:

$$P(Z_T|A, \pi) = \prod_t P(z_t|z_{t-1}, A)P(z_0|\pi). \tag{3}$$

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 straightforward using the standard forward-backward algorithm. See [9] for details.

3. Bayesian Network Classi£ers

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 dif£cult 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 [13]. 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 classi£cation task can result in poor performance, as described in [6]. To illustrate this point, consider a dataset of N records $D = \{x_1, ..., x_N\}$. Let $x_i = \{s_i, y_i\}$, where s_i is the classi£cation node (e.g. the speaker node in Figure 1) and y_i is the set of observations for record i. Substituting into Eqn 1 we have

$$\begin{aligned}\theta^* &= \arg\max_\theta \prod_i P(s_i, y_i|\theta)P(\theta) \\ &= \arg\max_\theta \sum_i (\log P(s_i|y_i, \theta) + \log P(y_i|\theta) + \log P(\theta))\end{aligned} \tag{4}$$

The classi£cation performance of the network is governed by the £rst term in Eqn 4, known as the conditional log likelihood. Since the parameter estimate maximizes the total posterior, it is not guaranteed to give an optimal estimate for the conditional likelihood under the structure B_s. If the structure is incorrect, the resulting classi£er may not generalize well during testing. Unfortunately, it does not seem to be possible to extend the closed form solutions for Eqn 4 to the conditional likelihood term in isolation. See [6] for details.

One approach to this problem is to use boosting to improve the classi£cation performance. In the Adaboost algorithm of Schapire et. al. [18], performance is improved by linearly combining a sequence of weak classi£ers, each of which is trained to correct the mistakes of the previous one on the training data. In previous work [14], we used boosted parameter learning to improve the speaker detection performance of the DBN classi£er of Figure 2. Boosting modi£es the parameter estimates by changing the weights on the training data according to the classi£er's performance. This approach is attractive because it can utilize ef£cient parameter learning algorithms for DBNs and the computational cost is a constant multiple of the cost without boosting.

Boosting has a particularly simple interpretation for discrete variable networks such as Figure 2. For simplicity, consider just the classi£er node s (i.e. speaker) and the measurements y. Boosting modi£es B_ϕ 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 classi£cation given by $P(s = s_i|y = y_j)$ is incorrect or correct. We can write this as $B_\phi = \mathrm{Histogram}(D_k, B_c)$.

Similarly, boosting will modify $\{A, \pi\}$ according to $P(s_0|D_k)$ and $P(s_t|s_{t-1}, D_k)$. We can write $\{A, \pi\} = \mathrm{Histogram}(D_k, B_d)$. Intuitively, boosting will increase or decrease the weighted counts for a pair of state transitions (A_{ij}, A_{jk}) based on the classi£cation performance for $s_t = j$. 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 classi£er of Figure 2.

Consider the decision boundary between $(s = -1, y = y_j)$ 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 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 £nal classi£er produced by averaging the reweighted classi£ers may not give signi£cant improvement in the discrete case. In the next section, we introduce an alternative combination scheme called *max-select*, in which we select the output of the classi£er which maximizes the likelihood of the test probe.

4. Learning Network Structure

The network structure in Figure 2 was manually speci£ed using knowledge about the problem and sensors. Manual design may introduce unwanted bias, and will be dif£cult for more complicated networks with many features. An alternative would be to learn the network structure automatically from the data [3, 8, 10]. Structure learning algorithms accomplish this by searching over the space of network structures to £nd the structure which is best-supported by the data. This requires a scoring function for candidate structures and an ef£cient search procedure, since the space of all topologies is intractably large for even a small number of nodes.

A Bayesian scoring function $H(B_1, B_2)$, for two candidate structures B_1 and B_2, can be constructed from the ratio of their posterior probabilities:

$$H(B_1, B_2) = \frac{P(B_1|D)}{P(B_2|D)} = \frac{P(D|B_1)P(B_1)}{P(D|B_2)P(B_2)}, \tag{5}$$

where D is the dataset and the last equality follows from Bayes Rule. If the prior over the structure is uniform, then the scoring function will reward the structure that maximizes the likelihood of the data.

As in the parameter learning case, it is quite possible that the maximal structure does not result in the best-performing classifier, since the scoring function in Eqn 5 is unsupervised. We now describe a novel boosting algorithm for structure learning which addresses this limitation. Figure 3 gives the flow chart for the algorithm. The key point is that Eqn 5 can be easily modified to operate on weighted data. As in the discrete paramter learning case, the effect of Adaboost from the standpoint of the structure learning module is simply to increase or decrease the frequency of different combinations of variables in the training data.

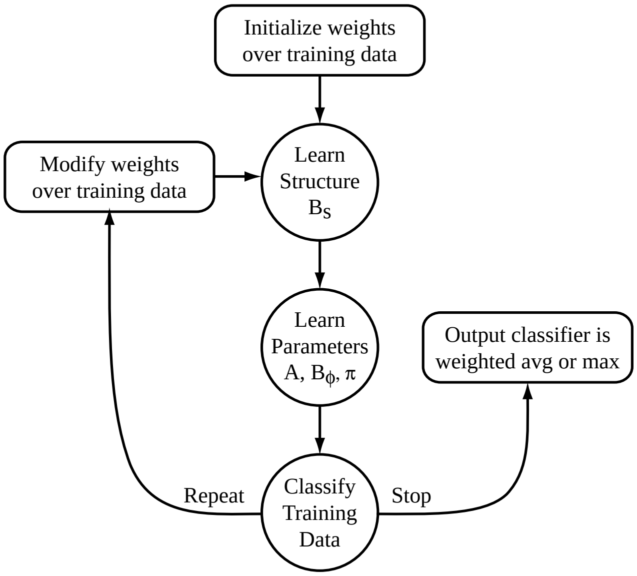
Initialize weights over training data
Modify weights over training data
Learn Structure B_S
Learn Parameters A, B_ϕ, π
Output classifier is weighted avg or max
Repeat
Classify Training Data
Stop

Figure 3. Flowchart for boosted structure learning algorithm.

Any standard structure learning algorithm can be used for the module in Figure 3. In this paper we use an MCMC variant of the K2 structure learning algorithm by Cooper and Herskovitz [3], which is described in [7]. K2 is a greedy search algorithm which uses a known ordering of the nodes and maximum limit on the number of parents for any node to constrain the search over network structure. The MCMC variant samples from the space of node orderings and uses K2 to compute the best structure for each sample. This approach retains the relative efficiency of K2 while ameliorating the constraints of node ordering.

Following the structure learning stage in Figure 3, parameter learning is performed in the current structure B_s. Parameter learning is also based on weighted data, using the procedure from our earlier work [14]. The result is a new classifier, which is then used to classify the training data so that the weights can be modified for the next iteration. The pseudocode for boosted structure learning follows:

```
Training
Given: $D = \{(s_1, y_1), \ldots, (s_T, y_T)\}$, with $s_t \in \{-1, +1\}$,
$y_t \in Y_T$ a sequence of $T$ data records;
Initialize $P_D^0(t) = 1/T,\ t = 1 \ldots T$;
For $i = 1 \ldots N$,
    $B_s^i = \mathrm{LearnStructure}(D,\ P_D^{i-1})$;
    $\theta^i = \mathrm{LearnParameters}(D,\ P_D^{i-1},\ B_s^i)$;
    $\{\alpha^i, P_D^i\} = \mathrm{Reweight}(D,\ P_D^{i-1},\ B_s^i,\ \theta^i)$;
end;

Testing
Given test data $\mathcal{Y}_T$, evaluate $N$ classifiers:
    $\hat{s}_t^i = \arg\max_j P(s_t = j \mid \mathcal{Y}_T,\ B_s^i,\ \theta^i),\ \ i = 1 \ldots N$;
The combined classifier output is:
    $\hat{s}_t = \mathrm{sign}(\sum_i \alpha^i \hat{s}_t^i)$; (*weighted average*)
    $\hat{s}_t = \hat{s}_t^k, k = \arg\max_i P(y_t | \hat{s}_t^i, B_s^i,\ \theta^i)$; (*max-select*)
```

```
Function $B_s = \mathrm{LearnStructure}(D,\ P_D)$
    Local var $L^*$, initially $L^* = 0$;
    Repeat
        $I$ = sampled node order for the static network;
        $\{B_c,\ L\} = \mathrm{K2}(D,\ P_D,\ I)$;
        Accept or reject $B_c$ given $L, L^*$, proposal dist;
    Until accept;
    $L^* = L$;
Return $\{B_c,\ B_d\}$;

Function $\theta = \mathrm{LearnParameters}(D,\ P_D,\ B_s)$
    $B_\phi = \mathrm{Histogram}(D,\ P_D,\ B_c)$;
    $\{A,\ \pi\} = \mathrm{Histogram}(D,\ P_D,\ B_d)$;
Return $\{B_\phi,\ A,\ \pi\}$;

Function $\{\alpha^k,\ P_D^k\} = \mathrm{Reweight}(D,\ P_D^{k-1},\ B_s,\ \theta)$
    $\hat{s}_t = \arg\max_j P(s_t = j \mid Y_T,\ B_s,\ \theta)$, for $t = 1 \ldots T$;
    $\varepsilon^k = \sum_t P_{\{t \sim P_D^{k-1}\}}(\hat{s}_t \neq s_t)$;
    $\alpha^k = \frac{1}{2} \log \left( \frac{1-\varepsilon^k}{\varepsilon^k} \right)$;
    $P_D^k(t) = \begin{cases} \frac{P_D^{k-1}(t) \exp \alpha^k}{Z^k} & \text{if } \hat{s}_t \neq s_t, \\ \frac{P_D^{k-1}(t) \exp -\alpha^k}{Z^k} & \text{if } \hat{s}_t = s_t. \end{cases}$ for $t = 1 \ldots T$;
Return $\{\alpha^k,\ P_D^k\}$;
```

	Speaker		Chess	
Static BN	Det	Delta	Det	Delta
Fixed Struct, AdaBoost	69%	0%	88%	0%
Fixed Struct, Max select	71%	2%	92%	4%
Learn Struct, No boost	87%	16%	94%	6%
Learn Struct, AdaBoost	87%	16%	94%	6%
Learn Struct, Max select	88%	17%	96%	8%
Dynanic BN				
Fixed Struct, No boost	80 %	9%		
Fixed Struct, AdaBoost	90 %	19%		
Learn Struct, No boost	90 %	19%		
Learn Struct, AdaBoost	91 %	20%		
Learn Struct, Max select	92 %	21%		

Table 1. Results for BN classification

5. Experimental Results

We have conducted experiments on two separate datasets: (i) speaker detection dataset and (ii) the *Chess* dataset from the UCI machine learning repository [1]. The speaker detection dataset consists of five sequences of a user playing a blackjack game in the Genie Casino Smart Kiosk setup. The sequences are of varying duration (from 2000-3000 samples) totaling to approximately 10000 frames. Each DBN state in each frame was hand labelled. The chess dataset has 36 attributes and two output classes.

The experimental results for both datasets are summarized in Table 1. For both the speaker and chess datasets, we measured classification accuracy for the static Bayesian network with structure learning using no boosting, AdaBoost, and max-selection boosting. For the speaker dataset we also measured the classification accuracy for the dynamic Bayesian Network with and without boosting. For each combination we present the overall detection rate (the *Det* column) as well as the improvement over the baseline classifier (the *Delta* column). The baseline classifier in each case was a static BN with fixed structure and standard parameter learning.

We can make several observations about the results in Table 1. First, structure learning led to improved classifier performance on both datasets. In the static speaker case, for example, standard structure learning (row 3) yielded an improvement of 16% over the baseline, suggesting that the hand-designed structures we used in our earlier experiments had significant bias. Furthermore, conventional structure learning met or exceeded the performance of parameter boosting with manually-specified structure. This illustrates the importance of selecting the right structure in Bayesian network classifier design.

We now examine some of the speaker-detection networks that were produced by structure learning. The static speaker network (B_c) shown in Figure 4 is the result of applying standard K2 to the node ordering given in the manually specified structure of Figure 1. This constraint results in qualitatively similar parent relationships between the manual and learned models, while allowing the data to determine the details. The network in Figure 4 yielded a classification accuracy of 78% on the testing set. While this is superior to the performance of Figure 1 (around 70%), it is below the best structure estimates in Table 1.

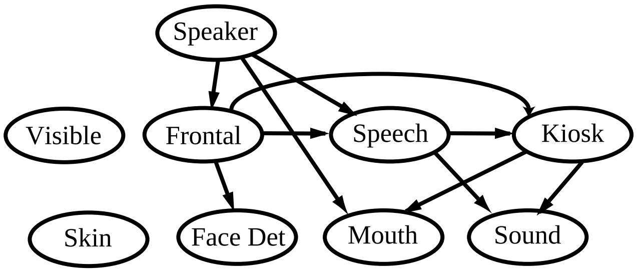

Figure 4. Structure obtained by K2 using the node ordering from Figure 1.

We can make several observations about Figure 4. First, it is clear that the nodes *visible* and *skin* are conditionally independent from the rest of the variables. This is a consequence of the training dataset, which did not contain any cases in which a visible face (and therefore skin pixels) were not present. As a consequence, the *visible* and *skin* variables did not provide any useful information for classification. The children of the *speaker* node (which is the output node during classification) are quite similar in both the learned and manual structure, as a consequence of the ordering constraint.

An intriguing property of the learned graph is the large number of connections between the *kiosk* node and other variables in the system. This node represents the state of the kiosk interface. In our Blackjack application, it encodes whether it is the user's turn to place a bet and whether or not any of the game-playing agents are talking to the user. As a result, it is a powerful source of information in speaker-detection. The next structure example makes this point even more clear.

Figure 5 shows a second network for static speaker detection. It was obtained using the MCMC K2 algorithm described in [7]. In contrast to Figure 4, this network does not respect the manually-specified node ordering.[1] This is the best static network we obtained, with a classification rate of 87%.

A striking feature of Figure 5 is the dominant role of the *kiosk* node. It has the most children (4) and has a causal

[1] Note that we have omitted the *visible* and *skin* nodes for clarity.

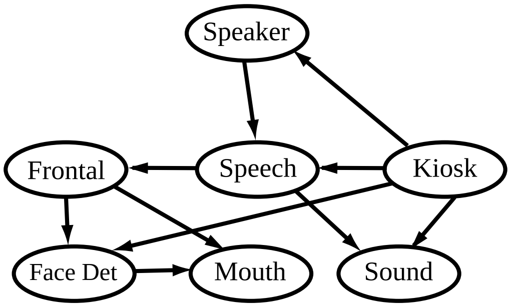

Figure 5. Structure produced by MCMC K2.

effect on the entire network. Sensitivity analysis reveals that the *kiosk* node is the most significant piece of evidence available to the classifier. While we guessed that this would be a valuable cue, our hand-specified models significantly underestimated its importance to the overall network. This is an example of the valuable insight that structure learning (and the graphical models formalism in general) can provide.

Boosting the structure learning process, as described in Section 4, yielded additional performance improvements in both the dynamic BN case (rows 9 and 10 in Table 1) and in the static case when max-selection was employed (row 5). It is curious that the gains from boosting structure learning were only a few percentage points, while the gains from parameter boosting were quite significant (e.g. 10% in the dynamic speaker case). We are currently working to clarify this result. Finally, on both datasets we found that our max selection rule led to improved performance over the standard AdaBoost algorithm.

6. Conclusion

We extend the AdaBoost algorithm for dynamic Bayesian networks to include structure learning. We demonstrate modest 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 provide some insights into the performance of parameter boosting and structure learning for the speaker-detection task. Our algorithms have been validated on a real world speaker detection dataset and the standard ”chess” dataset from the UCI repository.

References

[1] C. L. Blake and C. J. Merz. Uci repository of machine learning databases. Univ. of California at Irvine, 1998. http://www.ics.uci.edu/m̃learn/MLRepository.html.

[2] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[3] G. Cooper and E. Herskovitz. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, pages 309–347, 1992.

[4] R. Cutler and L. Davis. Look who’s talking: Speaker detection using video and audio correlation. In *Proc. IEEE Intl. Conf. on Multimedia Expo (ICME)*, New York, NY, 2000.

[5] J. W. Fisher III, T. Darrell, W. T. Freeman, and P. Viola. Learning joint statistical models for audio-visual fusion and segregation. In *Proc. Advances in Neural Information Processing Systems*, Denver, CO, 2000.

[6] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[7] N. Friedman and D. Koller. Being bayesian about network structure. In *Proc. 16th Conf. on Uncertainty in AI (UAI)*, 2000.

[8] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proc. Conf. on Uncertainty in AI (UAI)*, Madison, WI, 1998.

[9] A. Garg, V. Pavlović, and J. M. Rehg. Audio-visual speaker detection using dynamic bayesian networks. In *Proceedings of Fourth International Conference on Automatic Face and Gesture Recognition*, pages 384–390, Grenoble, France, March 28-30 2000.

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

[11] S. Intille and A. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks. In *CVPR ’98 Workshop on Interpretation of Visual Motion*, 1998. Also see MIT Media Lab TR 454.

[12] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer-Verlag, New York, NY, 1996.

[13] D. Koller and A. Pfeffer. Object-oriented bayesian networks. In *Proc. of the 13th Conf. on Uncertainty in AI*, pages 302–313, Providence, RI, Aug 1997.

[14] V. Pavlović, A. Garg, J. M. Rehg, and T. Huang. Multimodal speaker detection using error feedback dynamic bayesian networks. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, volume 2, pages 34–41, Hilton Head, SC, June 13-15 2000.

[15] J. M. Rehg, M. Loughlin, and K. Waters. Vision for a smart kiosk. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 690–696, San Juan, Puerto Rico, June 17-19 1997.

[16] J. M. Rehg, K. P. Murphy, and P. W. Fieguth. Vision-based speaker detection using bayesian networks. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, volume 2, pages 110–116, Ft. Collins, CO, June 1999.

[17] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

[18] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.