Boosted Learning in Dynamic Bayesian Networks for Multimodal Speaker Detection

ASHUTOSH GARG, MEMBER, IEEE, VLADIMIR PAVLOVIĆ, MEMBER, IEEE, AND JAMES M. REHG

Invited Paper

Bayesian network models provide an attractive framework for multimodal sensor fusion. They combine an intuitive graphical representation with efficient algorithms for inference and learning. However, the unsupervised nature of standard parameter learning algorithms for Bayesian networks can lead to poor performance in classification tasks. We have developed a supervised learning framework for Bayesian networks, which is based on the Adaboost algorithm of Schapire and Freund. Our framework covers static and dynamic Bayesian networks with both discrete and continuous states. We have tested our framework in the context of a novel multimodal HCI application: a speech-based command and control interface for a Smart Kiosk. We provide experimental evidence for the utility of our boosted learning approach.

Keywords*—*Bayesian networks, boosting, discriminative learning, multimodal integration, speaker detection.

I. INTRODUCTION

Human-centered user interfaces based on vision, speech, and other communication modalities present challenging sensing problems in which multiple sources of information must be integrated to infer the user's actions and intentions. A comprehensive framework for multimodal sensory integration should exhibit two key properties. It should: 1) combine information from noisy and ambiguous sensors with expert knowledge and contextual information; and 2) support tractable and accurate multimodal inference. Statistical techniques which combine human knowledge and data-driven learning can, therefore, play a critical role in multimodal system design.

Bayesian network (BN) [23], [13] models are an attractive framework for statistical modeling, as they combine an intuitive graphical representation with efficient algorithms for inference and learning. BNs encode conditional dependence relationships among a set of random variables in the form of a graph. An arc between two nodes denotes a conditional dependence relation, which is parameterized by a conditional probability model. Section III-A contains an illustration of a BN which fuses three vision-based sensors to estimate head pose. The structure of the graph encodes domain knowledge, such as the relationship between sensor outputs and hidden states, while the parameters of the conditional probability models can be learned from data. None of the simple vision sensors in the example inSection III-A were originally designed to analyze head pose, but they can be combined to form a more complex head pose sensor using the BN framework. Another advantage of the BN formalism is that it can be easily extended to handle time series data, by means of the dynamic BN (DBN) framework. Temporal modeling is very important for many audio and video fusion tasks.

The challenge in applying BN and DBN models to multimodal detection is twofold. First, the model has to accurately encode dependency relationships between multiple modalities that have been observed and identified by domain experts. In addition, the parameters that encode the detail of these dependency relations must be "refined" and reestimated from data in a particular multimodal context. This may mean that model parameters adapt to a particular noise level in the environment or that the strength of audiovisual dependency changes according to the contextual state of an environment. The classical BN learning paradigm can address modeling issues through an established methodology of parameter and structure learning techniques.

The second challenge in developing BN-based classifiers is the need to develop effective discriminative learning algorithms. In classical maximum-likelihood (ML) learning, model parameters are adjusted to achieve the best fit to a set of training data. Unfortunately, there is no guarantee that a model obtained in this fashion will make optimal classification decisions. A number of alternative discriminative learning approaches that directly attempt to optimize recognition accuracy have been considered in the past. For

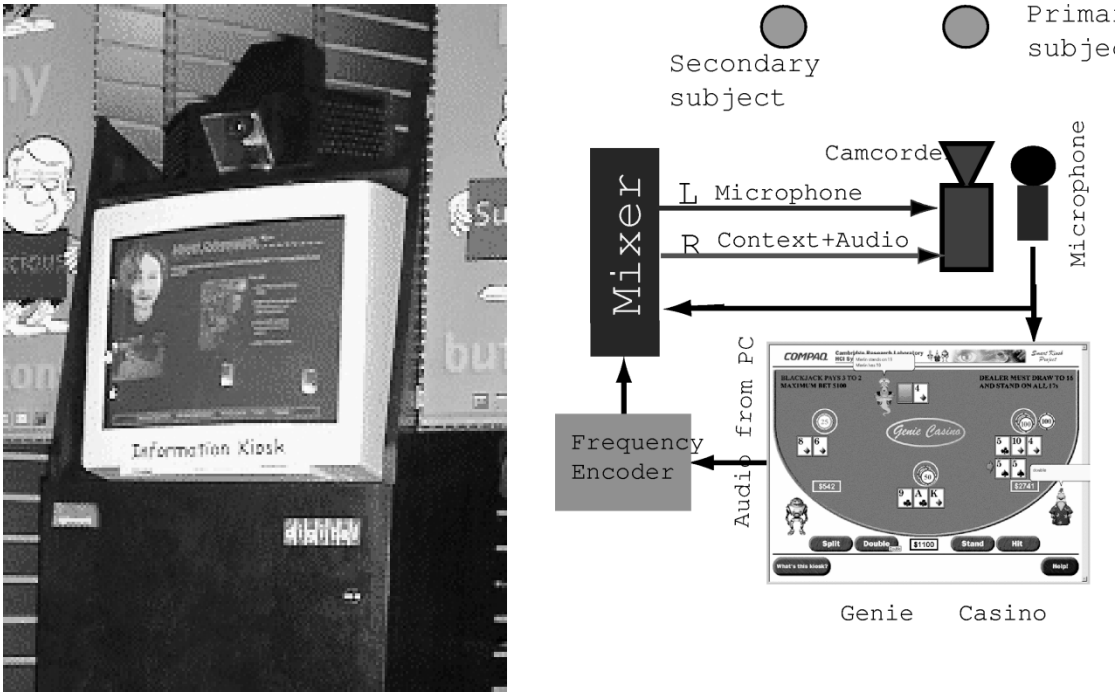
Manuscript received November 5, 2002; revised March 15, 2003.

A. Garg is with the IBM Almaden Research Center, San Jose, CA 95120 USA (e-mail: ashutosh@us.ibm.com).

V. Pavlović is with the Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA (e-mail: vladimir@cs.rutgers.edu).

J. M. Rehg is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: rehg@cc.gatech.edu).

Digital Object Identifier 10.1109/JPROC.2003.817119



(a) (b)

Fig. 1. The CRL Smart Kiosk. (a) Illustration of the Smart Kiosk as an information retreival booth. (b) Genie Casino interface and data collection setup.

instance, Bahl *et al.* [1] have suggested a method called "corrective training" for improving the performance of a speech recognition system. The method iteratively adjusts the parameter values to make the correct words more probable while reducing the probability of the incorrect ones. Similarly, [33] used a minimum classification error paradigm to build a speaker identification system and reported an improvement of 20% to 25% over a baseline ML model.

In this paper, we explore the use of boosting techniques to develop discriminative parameter learning algorithms for BN and DBN models. Boosting belongs to a family of machine learning techniques for *ensembles of classifiers* [5], [21], in which a set of classifiers is combined in making a decision. Most of the ensemble techniques rely either on averaging the outputs of the various classifiers, or selecting one of the classifiers from the pool depending on the input data. For instance, *boosting* [29] iteratively trains a series of classifiers that concentrate on errors made by their predecessors, thereby improving overall recognition performance. *Hierarchical mixture of experts* [15], on the other hand, makes use of a gating network to determine the weights of individual classifiers which are then linearly combined to yield the final classification decision. This method requires joint learning of parameters of the gating network and the classifiers, which may not be feasible in high dimensional spaces. Unfortunately, even though ensemble techniques are designed to improve the classification error, they sometimes yield small performance gains. One reason for this is that they often disregard the correlation among the outputs of individual classifiers.

In this paper, we describe a comprehensive BN framework for integration of multimodal sensory information. We demonstrate that BNs, as a class of graphical probabilistic models, have the representational strength to fuse data from weak sensors and encode dependency relationships between multiple modalities, both statically and dynamically. We show how such dependencies can be recovered and reestimated from data. In particular, we show that DBNs can optimally combine "off-the-shelf" video and audio sensors in solving a complex classification problem. We then describe a novel learning algorithm for DBNs that uses boosting to improve multimodal prediction accuracy. We refer to these new models as the *error feedback* BN models—error feedback BN (EFBN) and error feedback DBN (EFDBN). A variation of Freund and Schapire's Adaboost algorithm [30] is used to estimate the model parameters. A linear network is used to combine individual classifiers (here BNs and DBNs) into an EFBN or EFDBN. While each individual classifier is not trained to minimize the training error of the multimodal decision, the boosted combination achieves this goal. We demonstrate the utility of these new learning approaches in the context of a novel application domain: multimodal speaker detection for a Smart Kiosk user interface. In this domain, video and audio sensing support a command-based user interface which should be robust to multiple users and unconstrained environments. Our experiments show superiority of our model over the conventionally trained DBNs as well as static BNs. On our speaker detection task, we achieve an accuracy of 90%. These promising results suggest the general applicability of our methodology to other domains in which BN and DBN models are used. Preliminary versions of this work appeared in [8], [9], [22], [27].

II. Testbed: Multimodal Speaker Detection

Speaker detection is an important component of open-microphone speech-based user interfaces. For any interface

which relies on speech for communication, an estimate of the user's speaking state (i.e., whether or not he/she is speaking to the system) is important for its reliable functioning. In an open-microphone 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], [11]. Visual cues can be useful in deciding whether a user is facing the system and whether they are moving their lips. However, visual cues alone cannot easily distinguish a speaker from an active listener, who may be smiling or nodding without saying anything. Audio cues, on the other hand, can provide useful evidence for speech production. However, simple audio cues cannot easily distinguish a user who is speaking to the system from someone who is addressing their neighbor. Finally, contextual information describing the "state of the world" also has a bearing on when a user is speaking. For instance, in certain contexts, the user may not be expected to speak at all. Hence, the ideal solution would be to fuse audio, visual, and contextual cues in order to detect the presence of a speaker.

The CRL Smart Kiosk [3], [26], which was developed at the Cambridge Research Lab (CRL) of Compaq Computer Corporation, provides the context for our research on speaker detection. The Smart Kiosk is designed to provide information and entertainment to multiple users in public settings such as a shopping mall, subway station, etc. The kiosk has a speech-based interface which allows users to interact with the system using spoken commands. The public, multiuser nature of the kiosk application domain makes it ideal as an experimental setup for speaker detection task. The kiosk [see Fig. 1(a)] has a camera mounted on the top that provides visual feedback. A microphone is used to acquire speech input from the user while the audiovisual feedback for the user is provided in the form of a graphical avatar.

The experimental results in this paper are based on a specific instantiation of the Smark Kiosk known as the Genie Casino. In this instance, the kiosk implements a multiplayer blackjack game in which users are playing against the house in conjunction with other player avatars. Fig. 1(b) shows a screen shot from the Genie Casino interface. A command-and-control speech interface based on Microsoft SAPI allows a user to issue verbal commands such as "hit" or "stand," thereby controlling the play of their hand.

A. Sensors

Audio and visual information can be obtained directly from the two kiosk sensors (camera and microphone). We use a set of five "off-the-shelf" visual and audio sensors: the CMU face detector [28], a Gaussian skin color detector [35], a face texture detector, a mouth motion detector, and an audio silence detector. These components have the advantage of either being easy to implement or easy to obtain, but they have not been explicitly tuned to the problem of speaker detection. We now give a brief description of the sensors used for this problem.

1) Skin Sensor: Skin detection is a standard method for identifying skin-colored regions in an image. These regions can identify the location of a user's face and provide an attentional mechanism for visual tracking algorithms. It has been shown that skin pixels form a compact set in color space under general lighting conditions [14], and under restricted lighting conditions, the skin-color distribution can be characterized by a multivariate Gaussian [35]. We have implemented a simple Gaussian color model and applied it to skin detection. The skin sensor takes a part of the image as an input and outputs a scalar value which represents the presence or absence of skin in the image region. The percentage of the pixels in the image which belong to the skin color forms a measure for this sensor.

2) Texture Sensor: There are many objects, such as walls and doors, whose color is similar to skin. A simple texture feature is designed to help discriminate regions containing faces from regions containing either very smooth patterns such as walls or highly textured patterns such as foliage. A simple correlation ratio of the image window $g(x, y)$ defines the feature

$$T = \frac{E[g(x,y) \cdot g(x+\tau, y)]}{E[g^2(x,y)]} \tag{1}$$

where τ is a parameter which can be varied depending upon the smooth and coarse regions.

3) Face Sensor: By aligning the video camera with the display, we can ensure that users who are facing the kiosk will generate frontal, upright faces in the camera images. We search for these faces using the neural network-based Carnegie Mellon University (CMU),Pittsburgh, PA, face detector [28]. Since this detector is computationally expensive, we have built a linear Kalman filter model which keeps track of the scale and position of the face in the previous frame, and a region (spatial and scale) around that is used for searching the face in the next frame. The output of this sensor is highly saturated and pose sensitive. The tracker is capable of automatic initialization any time the face becomes lost.

4) Mouth Motion Sensor: An approximate optical-flow-based approach is used to measure the motion in the mouth region of a face image. The sensor uses three images from a video sequence, and after affine stabilization to compensate for head motion, the sensor outputs the residual error in the mouth region. Affine warping is used to cancel small global facial movements, and then the residual over the mouth region is normalized with respect to the residual over the remainder of the face. In the absence of correct mouth segmentation, the detector results are sensitive to head rotation. As the face pose approaches the profile view, residuals around the occluding contour increase, biasing the sensor.

5) Silence Detector: We employ a simple energy-based silence detector to signal the presence of speech in the kiosk environment. The audio is sampled at 4 kHz but then it is averaged over multiple frames to synchronize the audio output with the video. The energy of the signal is normalized using a sliding window, then a threshold is set which decides whether the environment contains any audio signal. The output of the sensor is sensitive to any fluctuations in the background noise.

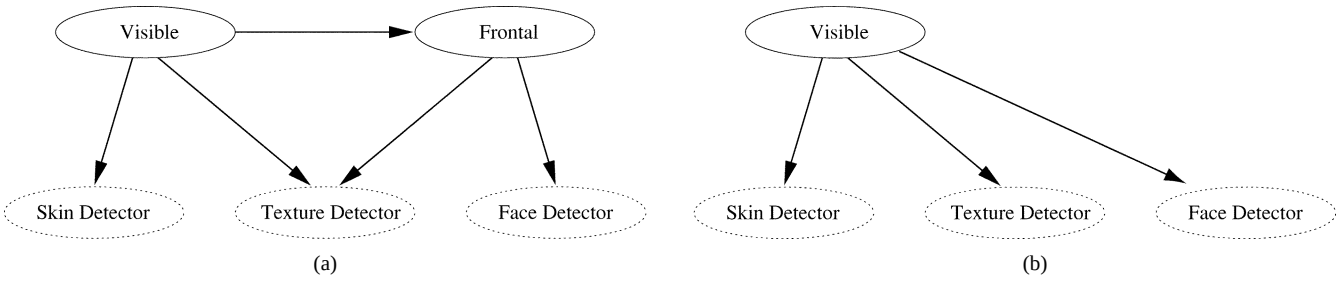


Fig. 2. (a) Vision network. (b) Simple naive Bayes network.

6) Contextual Information: In the case of the Genie Casino kiosk interface, we have complete access to the state of the interaction with the user, and this provides us with a sensor for contextual information. This sensor is of the utmost importance, since the blackjack game has a strong turn-taking structure which influences the speaker state. For example, when the dealer (the Genie) has asked the user to place a bet, the likelihood of the user speaking increases significantly. In contrast, while the dealer is speaking or attending to another player, the likelihood that the user is speaking decreases. In our implementation, we select a simplified state of the game as the contextual information. Namely, two game states are encoded: the user's turn (to interact) and its complement.

B. Experimental Setup

Fig. 1(b) shows the experimental setup for data collection. A camcorder is used to record video, audio, and contextual sensor data in a synchronized manner. In the Genie Casino system, a finite-state machine encodes the progress of the game. The game state is converted to a frequency-encoded audio signal, which is output through the speaker port. This signal is mixed with the single microphone input to provide a stereo input to the camcorder. In this way, all of the sensor data is synchronized in time and accessible using standard digitization tools.

III. Bayesian Networks for Speaker Detection

Speaker detection represents a challenging ground for testing the representational power of BN models in a complex multisensor fusion task. Different types of sensors need to be seamlessly integrated in a model that both reflects the expert knowledge of the domain and the sensors and benefits from the abundance of observed data. We approach the model-building task by first tackling the expert design of networks that fuse individual sensor groups (video and audio). We then proceed with the integration of these sensor networks with each other, with contextual information, and over time. Finally, the data-driven aspect comes into play with data-driven model learning.

A. Visual Network

The goal of the vision network is to infer whether a user is present (i.e., their face is visible in front of the kiosk), whether they are facing the kiosk, and whether their mouth is moving. To accomplish the first two tasks, we designed a small BN which fuses the output of the skin color, texture, and face detector sensors and estimates the visibility and orientation (frontal versus side) of the user's face. This network structure, which is an example of a polytree, is depicted in Fig. 2(a). The "visible" and "frontal" variables are not directly observed, but can be inferred from sensory data. The specific network topology results from our expert knowledge of the sensor properties. The user being "frontal" clearly depends on whether he is "visible." If the user is "visible," parts of his skin and face will appear in the image. On the other hand, the face detection sensor only detects frontal faces. Hence, it is plausible to connect it to the "frontal" node.

The arcs between variables qualify the known conditional dependencies. They are parameterized by conditional probability distributions that quantify those dependencies. The arc between the two binary variables "frontal" and "visible," for example, stores the two-by-two conditional probability table (CPT), $\Pr(\text{frontal}|\text{visible})$. In the rest of the paper, we will refer to a total set of all CPT parameters for a particular network by θ. The network topology will be denoted by B. Together, they form a tuple (B, θ) that fully specifies each network.

The probability distribution defined by the visual network is now

$$\Pr(V, F, \text{SK}, \text{TX}, \text{FD}) = \Pr(V)\Pr(F|V) \cdot \Pr(\text{SK}|V)\Pr(\text{TX}|V, F)\Pr(\text{FD}|F)$$

where V, F, SK, TX and FD correspond to the "visual," "frontal," "skin," "texture," and "face detector" nodes, respectively. If one wanted to use the visual network as a face detector, the posterior distribution of interest would be $\Pr(V|\text{SK}, \text{TX}, \text{FD})$. This posterior can be efficiently obtained using a number of BN inference techniques (e.g., junction tree potentials [23]). The optimal Bayesian decision that the face was present is then made using the likelihood ratio test

$$\log \frac{\Pr(V = \text{true}|\text{SK}, \text{TX}, \text{FD})}{\Pr(V = \text{false}|\text{SK}, \text{TX}, \text{FD})} > 1.$$

Consider the alternative face detector network illustrated in Fig. 2(b), which is an example of a naive Bayes classifier. In comparison to the vision network, this network would make a poor face detector because it fails to take into account the fact that the neural network face detector (the FD sensor) can only detect upright, frontal faces. If the face in the image is tilted or rotated by more than 10° to 15°, the

FD sensor will not respond to it. For example, consider the case where FD $= 0$ while SK $= 1$ and TX $= 1$. This could occur if the user has turned their head to one side to speak to their neighbor. Significant skin and texture cues would still be present, but the face detector would not register a face. In the case of the naive Bayes classifier of Fig. 2(b), these contradictory sensor readings would have the effect of reducing $\Pr(V = \text{true})$.

In the vision network, however, the sensor value FD $= 0$ can be *explained away* by inferring that $F =$ false while $V =$ true, effectively encapsulating the knowledge that the neural network cannot detect nonfrontal faces. Thus, by moving to a slightly more complex BN architecture, a polytree rather than a tree, we not only increase the classification accuracy for V but also infer the value of F without directly measuring it. The phenomena of explaining away is a key property of BN models for cue fusion. We can view the vision network as a simple pose detector which is constructed from off-the-shelf components, none of which explicitly measure the pose. The simplicity and moderate computational cost of this approach stands in contrast to the much more complex and costly procedures required for full-scale head tracking (see [20] for a representative example).

1) Discrete and Continuous Sensor Models: In the preceding discussion, we have indirectly implied the discrete nature of sensor outputs. All sensors were assumed to output binary decisions, such as SK $= 0$ (skin not detected) and SK $= 1$ (skin detected) for the skin sensor. This resulted in the nonparametric distribution model for $\Pr(\text{SK}|V)$ defined by a two-by-two CPT. However, our framework is not in any way restricted to a discrete sensor model. A parametric model such as a conditional Gaussian distribution can be used to model $\Pr(\text{SK}|V)$, $\Pr(\text{SK}|V) \propto \mathcal{N}(\text{SK}|\mu_V, \Sigma_V)$. Indeed, we will use such models to study the influence of sensor discretization in our application. As long as no instances occur where a parent of a discrete child is a continuous variable, the inference and learning in BNs are of equal complexity for discrete, continuous, and mixed variable models alike [23].

B. Audio Network

The role of the audio network, depicted in Fig. 3, is to infer whether speech production is occuring in the scene. The two sensors "mouth motion" and "silence detector" provide complementary cues about speech production. The silence detector measures any sudden change in the ambient audio environment that would signal the onset of talking. Since there are many possible sources of audio energy (from background speech to copiers/printers), the mouth motion sensor provides an additional test. An increase in audio energy which is correlated to mouth motion provides strong evidence for talking. The probability distribution defined by the audio network is simply

$$\Pr(A, \text{MM}, \text{SL}) = \Pr(A)\Pr(\text{SL}|A)\Pr(\text{MM}|A)$$

where A, MM and SL denote "audio," "mouth motion," and "silence" nodes, respectively.

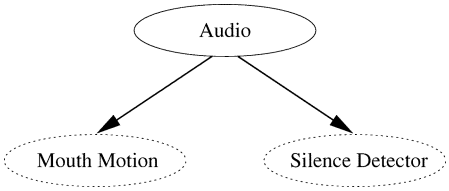


Fig. 3. Audio network for speaker detection.

C. Integrated Audiovisual Network

The integrated BN model combines the contextual sensor (which encodes the state of the blackjack game) with the visual and audio subnetworks described in the previous two sections. It is illustrated in Fig. 4. The "speaker" node comprises the output of the classifier (i.e., true if a speaker was detected and false if it was not). The chosen network topology represents our knowledge of the audio, visual, and contextual conditions that correspond to the presence of a speaker: in the course of the game when the user is expected to speak, they should be facing the kiosk and talking. The optimal classification decision is made by comparing the posteriors $\Pr(S = \text{true}|Y)$ and $\Pr(S = \text{false}|Y)$, where $Y = \{\text{SK}, \text{TX}, \text{FD}, \text{MM}, \text{CT}\}$ is the total set of sensor outputs.

D. Dynamic Network

The final step in designing the speaker detection architecture is the addition of a temporal component which links the values of nodes over time. This effectively converts the static BN of Fig. 4 into a dynamic BN. The temporal component captures our intuition that decisions about the speaker (as well as the frontal and audio states) should not change dramatically over short time scales. As we will see in Section VII, the dynamic component of the model significantly improves the classifier performance by smoothing the estimates over time and thereby eliminating rapid fluctuations in classifier decisions.

Fig. 5 illustrates the temporal structure of our DBN model. We introduce temporal dependencies between the three primary hidden states in the model. While individual sensors may in fact exhibit their own temporal structure, we have chosen to limit the temporal connections to the main hidden states in order to control the complexity of the model structure. The presence of all possible arcs among the three hidden state nodes results from our lack of exact knowledge about the structure of the temporal dependencies, i.e., we allow for all dependencies to be present and later on determine the specific parameters from the training data.

The final DBN model which incorporates all of the previous elements is illustrated in Fig. 6. Here, the nodes outlined by dotted lines are the direct observation nodes while the once shown in solid are the unobserved nodes. The speaker node (see Fig. 4) is the query node for the classifier. The probability distribution encoded by this network can be written as shown in the equation at the bottom of the next page. This distribution is defined by the parameters of a Markov model: a matrix Π of transitions probabilities among states in Fig. 5 and their initial state

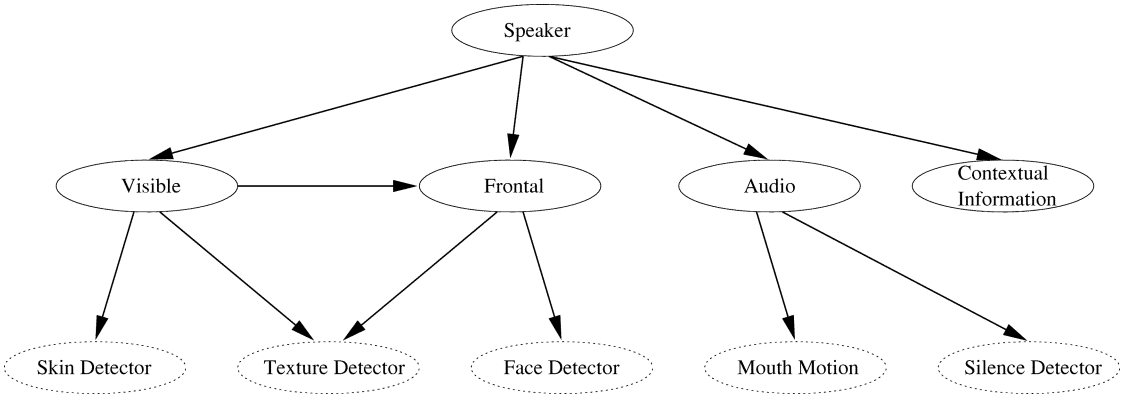


Fig. 4. Integrated audiovisual network.

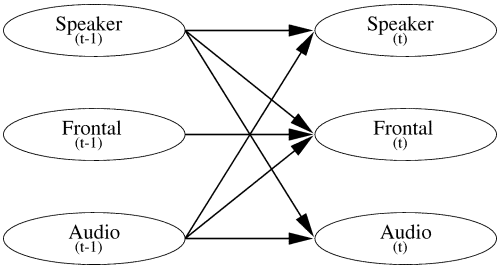


Fig. 5. Temporal dependencies between the speaker, audio, and frontal nodes at two consecutive time instances.

distribution π. In addition, the distribution of the integrated audiovisual network acts as the measurement model with CPT parameters B_ϕ.

Inference in this network now corresponds to finding the distribution of the speaker variable s_t at each time instance conditioned on a sequence of hexamodal (six-tuple) measurements from the sensors

$$\mathcal{M}_1^T = \{\mathrm{SK}_1, \mathrm{TX}_1, \mathrm{FD}_1, \mathrm{MM}_1, \mathrm{CT}_1, \mathrm{SD}_1, \ldots, \mathrm{SK}_T, \mathrm{TX}_T, \mathrm{FD}_T, \mathrm{MM}_T, \mathrm{CT}_T, \mathrm{SD}_T\}.$$

Optimal detection of the speaker at time t can now be made by comparing $\Pr(S_t = \text{true}|\mathcal{M}_1^T)$ to $\Pr(S_t = \text{false}|\mathcal{M}_1^T)$. These posteriors are obtained directly from the forward–backward inference algorithm (cf. [24]). One may also be interested in predicting the likelihood of the speaker from all the previous observations, $\Pr(S_{t+1} = \text{true}|\mathcal{M}_1^t)$.

With the topology of the models determined by expert knowledge of the problem domain, the related tasks of parameter learning and optimal inference are tackled in the following sections.

IV. Maximum-Likelihood Parameter Learning

We can describe the DBN model of Fig. 6 as the tuple (B_s, θ), where B_s encodes the structure (i.e., the topology) of the network and $\theta = \{B_\phi, \Pi, \pi\}$ is the set of network parameters. In this case, B_s has been specified manually. The parameters can be learned from a training data set D by computing

$$\theta^* = \arg\max_\theta P(D|\theta)P(\theta) \tag{2}$$

where $P(\theta)$ is a prior.[1] When all of the nodes are observed, this computation can be done in closed form [10].

Parameter learning is particularly simple for the network of Fig. 6. Let z denote the four hidden states and y denote the six measurements. Let $Z_T = \{z_1, z_2, \ldots, z_T\}$ be the sequence of T hidden states and X_T the corresponding sequence of measurements. Then we have

$$P(Z_T, Y_T, \theta) = P(Y_T|Z_T, B_\phi)P(Z_T|\Pi, \pi). \tag{3}$$

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

$$P(Z_T|\Pi, \pi) = \prod_{t=2}^{T} P(z_t|z_{t-1}, \Pi, \pi). \tag{4}$$

[1]Throughout this work we have employed simple noninformative Dirichlet priors to all discrete variables, which result in addition of equal, small pseudocounts in all CPT estimates. No priors were used for parameters of continous sensor models.

$$\Pr(\mathrm{SK}_1, \mathrm{TX}_1, \mathrm{FD}_1, \mathrm{MM}_1, \mathrm{CT}_1, \mathrm{SD}_1, V_1, \mathrm{SP}_1, F_1, A_1, \ldots \mathrm{SK}_T, \mathrm{TX}_T, \mathrm{FD}_T, \mathrm{MM}_T, \mathrm{CT}_T, \mathrm{SD}_T, V_T, \mathrm{SP}_T, F_T, A_T) =$$
$$\Pr(\mathrm{SP}_1, F_1, A_1) \Pr(\mathrm{SK}_1, \mathrm{TX}_1, \mathrm{FD}_1, \mathrm{MM}_1, \mathrm{CT}_1, \mathrm{SD}_1, V_1|\mathrm{SP}_1, F_1, A_1)$$
$$\prod_{t=2}^{T} \Pr(\mathrm{SK}_t, \mathrm{TX}_t, \mathrm{FD}_t, \mathrm{MM}_t, \mathrm{CT}_t, \mathrm{SD}_t, V_t|\mathrm{SP}_t, F_t, A_t) \Pr(\mathrm{SP}_t, F_t, A_t|\mathrm{SP}_{t-1}, F_{t-1}, A_{t-1}).$$

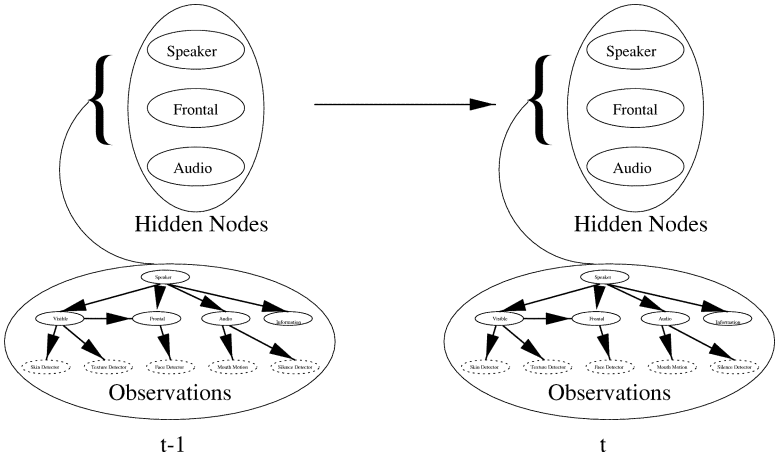


Fig. 6. Two time slices of the dynamic BN for speaker detection.

Thus, the transition matrix Π 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.

V. Discriminative Parameter Learning and Bayesian Network Classifiers

DBN models are an appealing framework for complex inference problems because they are interpretable, composable, and generative. *Post hoc* 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 BN models by combining subgraphs. This makes it possible to reuse and extend modeling components without retraining an entire model for each new problem [17]. Third, because the BN models a joint probability distribution, sampling can be used to generate synthetic data. This is another source of insight into network performance.

However, standard ML BN learning techniques can make them ill-suited for classification tasks such as speaker detection [7]. Parameter learning in a DBN is an example of a density estimation problem in which all of the variables in the model are treated equally. To understand how this could result in poor classification performance, consider a dataset of N records $D = \{x_1, \ldots, x_N\}$. Let $x_i = \{s_i, y_i\}$, where s_i is the classification node (e.g., the speaker node in Fig. 4) and y_i is the set of observations for record i. Substituting into (2), we have

$$\begin{aligned}\theta^* &= \arg\max_\theta \prod_i P(s_i, y_i|\theta)P(\theta) \\ &= \arg\max_\theta \sum_i [\log P(s_i|y_i, \theta) + \log P(y_i|\theta)] + \log P(\theta).\end{aligned} \tag{5}$$

The classification performance of the network is governed by the first term in (6), known as the conditional log likelihood. Since the parameter estimate maximizes the joint likelihood, it is not guaranteed to give an optimal estimate for the conditional likelihood under the structure B_s. Furthermore, 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 (6) to the conditional likelihood term in isolation. See [7], [12] for more details on this issue.

Our approach to the problem of discriminative learning is to use boosted ensembles of classifiers. In the Adaboost algorithm [29], [31], 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.

More formally, consider a binary classification problem with data given by $S = \{(s_1, y_1), \ldots, (s_m, y_m)\}$. Here, y_i is a feature vector (for instance, the hexamodal audiovisual feature), and s_i is the desired label (speaker state). The goal of the learning algorithm is to find a hypothesis (classifier) $h : Y \to S$ that minimizes misclassification. In a binary classification scenario, $s \in \{+1, -1\}$. The standard Adaboost algorithm is

Given: $D = \{(s_1, y_1), \ldots, (s_m, y_m)\}$, $y_i \in \mathcal{Y}$, $s_i \in \{-1, +1\}$;
Initialize distribution over data pairs $P_D^{(1)}(i) = 1/m$;
For $k = 1, \ldots, K$

- Train hypothesis h_k using data D with distribution $P_D^{(k)}$.
- Choose $\alpha_k = (1/2)\ln(1 - \epsilon_k/\epsilon_k)$ where $\epsilon_k = \Pr_{i \sim P_D^{(k)}}[h_k(y_i) \neq s_i]$
- Update:

$$P_D(k+1)(i) = \frac{P_D^{(k)}(i)\exp(-\alpha_k s_i h_k(y_i))}{Z_k}$$

where Z_k is the normalization factor.
The final hypothesis is

$$H(y) = \operatorname{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(y)\right)$$

Adaboost starts by assigning equal weight $P_D^{(1)} = 1/m$ to all the training data. In each iteration, the weight of the misclassified samples is increased whereas that of correctly classified samples is decreased—the algorithm "directs" the consecutive classifiers to focus on samples which were previously difficult to learn. The weight factor $P_D^{(k)}$ depends on the expected error made by the hypothesis h_{k-1}. The new hypothesis h_k is then trained with respect to those weights $P_D^{(k)}$. After T iterations, the final hypothesis is obtained as a linear combination of individual hypotheses.

Adaboost has a number of appealing properties. It can been shown that if the weights (α_k) are chosen in the way described previously than the training error is bounded by

$$\prod_k \left[2\sqrt{\epsilon_k(1-\epsilon_k)}\right]. \tag{6}$$

Hence, if the weak hypotheses h_k are slightly better than the chance, the training error decreases exponentially fast. Additional bounds on the generalization error can also be derived [29]. It has also been shown empirically that Adaboost has a good generalization property, unless the number of hypotheses becomes too large. While most arguments about the generalization property attribute it to the maximization by boosting of the between-class margin [6], some recent studies show that this can be explained by the additional smoothing introduced by multiple ensemble decisions [18]. Finally, note the two-decision algorithm can be extended to the case when s_i takes multiple values by using the multiple-class version of Adaboost [29].

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. The flowchart of the DBN parameter boosting algorithm is shown in Fig. 7.

Boosting has a particularly simple interpretation for the network of Fig. 6. For simplicity, consider just the classifier node s (i.e., speaker) and the measurements y. Boosting modifies B_ϕ according to the distribution $P(s|y, D_k)$ 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 Π according to $P(s_t|s_{t-1}, D_k)$. Intuitively, boosting will increase or decrease the weighted counts for a pair of state transitions (Π_{ij}, Π_{jk}) based on the classification performance for $s_t = e^j$ (i.e., s_t is in the jth state). This can be viewed as an error-driven duration density model for the Markov chain.

The boosting algorithm for a DBN is summarized as follows:

DBN Parameter Boosting

Training

Given: $D = \{(s_1, y_1), \ldots, (s_T, y_T)\}$, $s_i \in \{-1, +1\}$; $y_t \in Y_T$, a sequence of T data records;

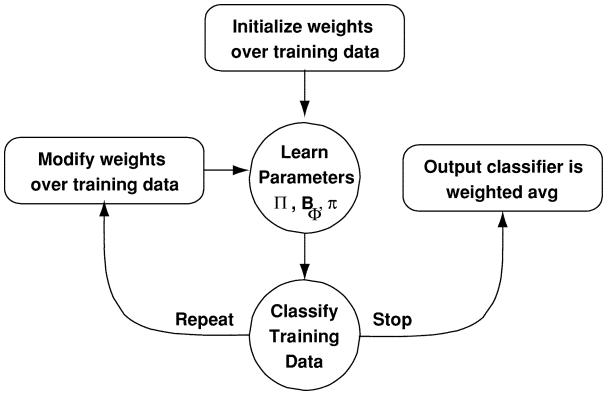


Fig. 7. Boosted parameter learning algorithm.

Initialize $P_D^0(t) = 1/T$, $t = 1, \ldots, T$;
For $i = 1, \ldots, N$,
$\theta^i = \text{LearnParameters}(D, P_D^{i-1}, B_s)$;
$\{\alpha^i, P_D^i\} = \text{Reweight}(D, P_D^{i-1}, B_s, \theta^i)$; end;

Testing

Given test data Y_T, evaluate N classifiers:

$$\hat{s}_t^i = \arg\max_j P(s_t = j|Y_T, B_s, \theta^i), \quad i = 1, \ldots, N;$$

The combined classifier output is:

$$\hat{s}_t = \text{sign}\left(\sum_i \alpha_i \hat{s}_t^i\right);$$

Function $\theta = \text{LearnParamaters}(D_t, P_D, B_s)$
$B_\phi = \text{Histogram}(D, P_D, B_c)$;
$\{\Pi, \pi\} = \text{Histogram}(D, P_D, B_d)$;
Return θ;
Function $\{\alpha^k, P_D^k\} = \text{Reweight}(D_t, P_D^{k-1}, B_s, \theta)$
$\hat{s}_t = \arg\max_j P(s_t = j|Y_T, B_s, \theta)$, $t = 1, \ldots, T$;
$\epsilon^k = \sum_t P_{t\ P_D^{k-1}}(\hat{s}_t \neq s_t)$;
$\alpha^k = (1/2)\log\left(1 - \epsilon^k/\epsilon^k\right)$;

$$P_D^k(t) = \begin{cases} \frac{P_D^{k-1}(t)\exp\alpha^k}{Z_k} & \text{if } \hat{s}_t \neq s_t \\ \frac{P_D^{k-1}(t)\exp -\alpha^k}{Z_k} & \text{if } \hat{s}_t = s_t \end{cases},$$

$t = 1, \ldots, T$;
Return $\{\alpha^k, P_D^k\}$;

The algorithm maintains a weight distribution defined over the data. It starts by assigning equal weight to all the samples. As the algorithm proceeds, the weight of correctly classified samples is decreased whereas that of misclassified ones is increased. Our observations show that the points where the error is made are normally the points which were classified with low confidence.

At each iteration, algorithm obtains an observation density matrix B_k using the present distribution over the data given by $P_D^{(k)}$. The DBN learning algorithm gives an estimate of the transition probability matrix Π, for which all the samples are considered to be equally probable. Once DBN is trained,

Fig. 8. Three frames from a test video sequence.

we use a DBN inference algorithm to decode the hidden state sequence. During decoding we obtain the most likely state, at any time, for the given observation sequence. This estimated state is compared with the true state, the discrepancy of which corresponds to an error. The final DBN model is somewhat akin to a mixture of DBNs—it uses the weighted sum of individual classifier decisions to yield an ensemble decision.

VI. Previous Work

A number of discriminative approaches to parameter learning in hidden Markov models (HMMs) have been developed by the speech recognition community. While this body of work is quite different from our boosted BN approach, it shares our goal of overcoming the poor classification performance of ML learning. Examples of discriminative HMM learning methods include corrective training [1] and minimum classification error training [16], [33].

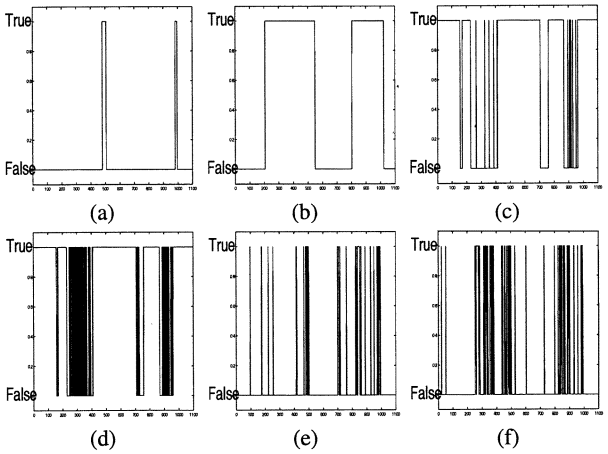


Fig. 9. (a) Ground truth for the speaker state, where "True" means that there is a speaker and "False" means an absence. The x axis gives the frame number in the sequence. (b) Contextual information, where "True" means it is the user's turn to play and "False" means the computer is going to play. (c), (d), (e), (f) Output of texture, face, mouth motion, and silence detector, respectively.

In contrast to the ensemble of classifiers produced by our boosting framework, these earlier discriminative techniques attempted to optimize the parameters of a single HMM classifier to achieve better generalization. This work lacks the theoretical guarantees on generalization which exist for boosting methods. Boosting is also considerably simpler and more general, since it requires only trivial modification of the base learning algorithm and can be applied to arbitrary BN and DBN models. A disadvantage of boosting in comparison to this prior work is the need to evaluate multiple classifiers during the testing phase.

The use of Adaboost to train a hybrid HMM/neural network speech recognizer was recently reported in [32]. The Adaboost algorithm was utilized to enhance the performance of the neural network measurement model, hence resulting in better overall recognition performance. In contrast, our method addresses the boosting of both static and dynamic model parameters within the same BN framework. Adaboost has also been applied to confidence scoring for an audio-indexing task [19].

Our application of boosted learning to audiovisual speaker detection is unique. However, there are a number of publications on related audiovisual detection problems. The majority of these works perform audiovisual fusion at the data level. For example, in [11], the mutual information between video and audio signals is the basis for the identification of joint audiovisual events. In [4], a time-delay neural network is used to detect correlations between audio and video at the signal level. In [34], a particle filter is used to fuse visual contour measurements and microphone array data for speaker tracking. The most relevant prior work is [2], which describes a BN model for audio and video signals which is used for speaker tracking. We differ from all of this previous work both in the level at which audiovisual modalities are fused

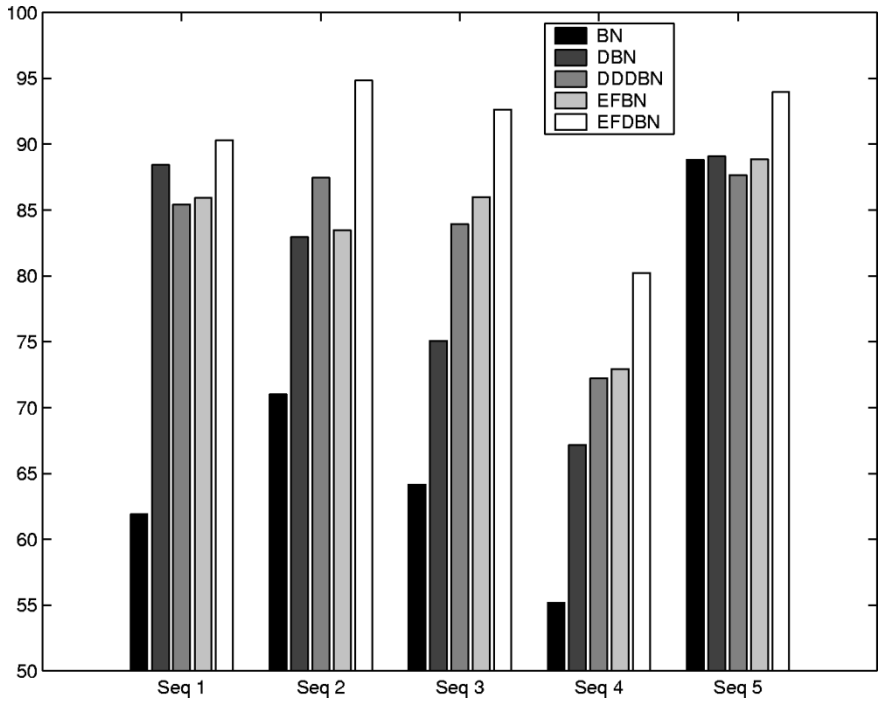


Fig. 10. Comparison of speaker prediction accuracy (%) obtained using different classification strategies. The chart shows that the performance of DBN is better that the static BN. The results show that EFDBN and EFBN outperform both static the BN and the DBN. EFDBN shows the highest gain in performance among all the studied models.

(the symbolic level) and in our use of boosting to improve classification performance.

VII. Experimental Results

We have conducted a series of experiments using a common data set to validate the modeling and predictive ability of our mulitimodal integration framework. The data set consists of five sequences of a user playing a blackjack game in the Genie Casino setup. The sequences were of varying duration (from 2000 to 3000 samples) totaling 12 500 frames. Fig. 8 shows some of the recorded frames from the video sequence.

Each sequence included audio and video tracks recorded through a camcorder along with frequency encoded contextual information [see Fig. 1(b)]. The visual and audio sensors were then applied to the synchronized audio and video streams.

Because some of the sensors provide continuous estimates of their respective functions (e.g., silence detector's internal output is the short-term energy of the audio signal), we considered two different modeling scenarios. In the first scenario, we *a priori* binarized measurements of all multimodal sensors. Decision thresholds were optimally determined for each sensor from distributions of sensory measurements to imply binary sensor states (e.g., silence versus nonsilence). These discretized states were then used as input for the DBN model. Examples of individual sensor decisions (e.g., frontal versus nonfrontal, silence versus nonsilence, etc.) are shown in Fig. 9. Abundance of noise and ambiguity in these sensory outputs clearly justifies the need for intelligent yet data-driven sensor integration. In the second modeling scenario, we retained "soft decisions" of the continuous sensors and used them directly in the integration architecture.

All detection measures were computed using fivefold cross validation.

A. Experiment Using Static BN

The first experiment was done using the static BN (given in Fig. 4) to form the baseline for comparison with the dynamic model. In this experiment, all samples of each sequence were considered to be independent of any other sample. Part of the whole data set was considered as the training data, and the rest was retained for testing. During the training phase, output of the sensors along with the hand label values for the hidden nodes (speaker, frontal, and audio) were presented to the network. The network does learn CPTs that are to be expected. The actual CPT values show that the presence of the speaker ($S = 1$) must be expressed through the presence of a talking ($A = 1$) frontal face ($F = 1$) in the appropriate context of the game ($C = 1$). On the other hand, the existence of the frontal face alone does not necessarily mean that the speaker is present ($S = 0$, $F = 1$).

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 (speaker, frontal, audio) is considered to be an error. Fivefold cross validation was used to split the dataset into training and test sets and evaluate the detection performance. An average accuracy of 70% is obtained (see Fig. 10 for results on individual sequences). The accuracy is low even though the learned network parameters do seem intuitive, as explained previously. Fig. 9 depicts a typical output of the sensors along with the ground

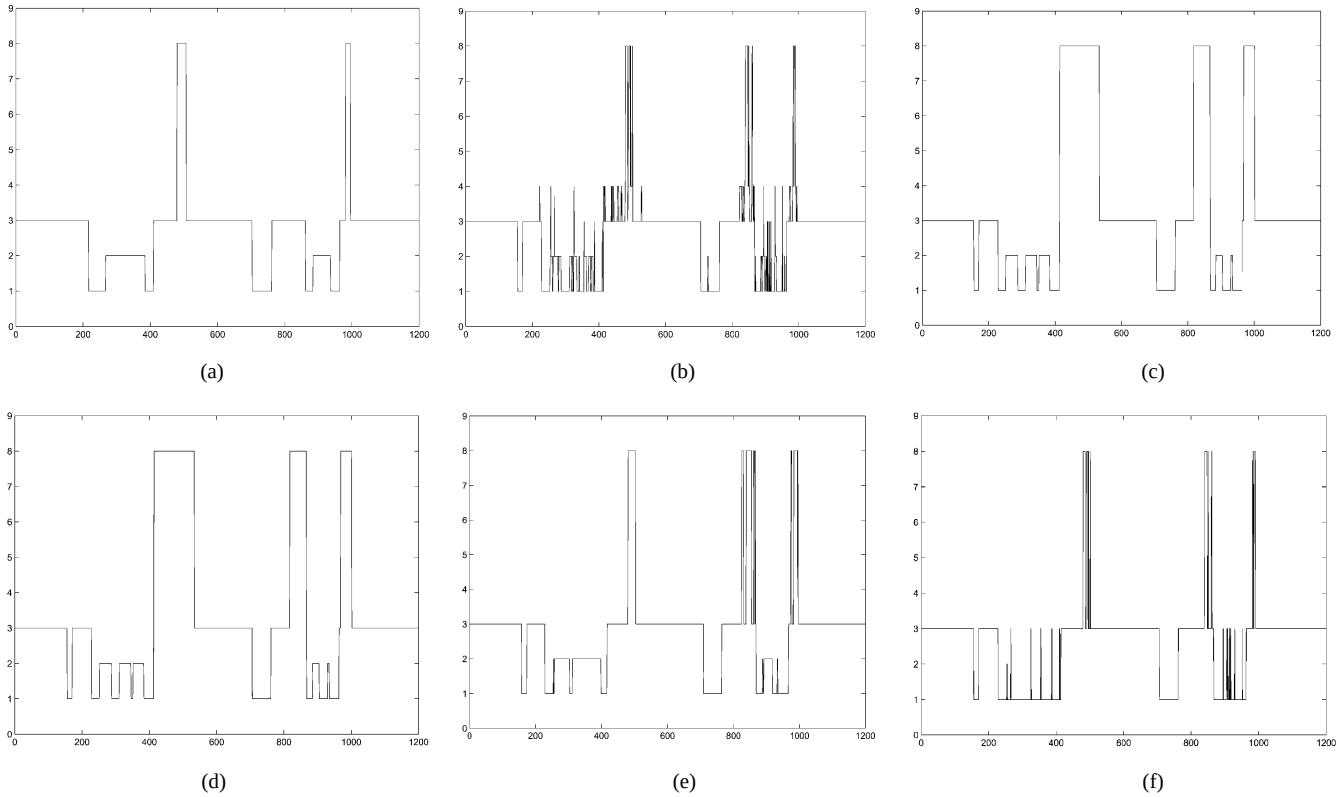


Fig. 11. (a) True state sequence. (b), (c), (d), (e), and (f) Decoded state sequences by static BN, DBN, DDDBN, EFDBN, and expert (formulated by knowledge of the system), respectively. (State 1—no speaker, no frontal, no audio; state 2—no speaker, no frontal, audio; state 3—no speaker, frontal, no audio; state 8—speaker, frontal, audio.).

truth for the speaker state. The sensor data is noisy, and it is hard to infer the speaker without making substantial errors. Fig. 11(a) shows the ground truth sequence for the state of the speaker, and Fig. 11(b) shows the decoded sequence using static BN. Nevertheless, because, on average, the speaker was present in 10% to 15% of all sequence frames, our detection rate is significantly above that of a random speaker detection. Note also that performance of the inference algorithm is especially poor on sequence 4. We attribute this to the excessive head motion and increased background noise present in this sequence, moreso than in the other four cases. The diminished performance on sequence 4 will be exhibited by all our inference algorithms, as will be seen in the sections later.

B. Experiment Using DBN

The next experiment was conducted using the DBN model. At sequence level, data was considered independent (e.g., seq1 is independent of seq2). The learning algorithm described in Section IV was employed to learn the dynamic transitional probabilities among frontal, speaker, and audio states. During the testing phase, a temporal sequence of sensor values was presented to the model, and Viterbi decoding (cf. [25]) was used to find the most likely sequence of the speaker states. Overall, we obtained the accuracy of the speaker detection (after cross validation) of about 80%, an improvement of 12% over the static BN model. An indication of this can be seen in actual decoded sequences.

Table 1
The Learned CPT for Transition Probabilities in DBN Model

$(S,F,A)_{t-1}$	$Pr(S_t=0,1)$		$Pr(F_t=0,1)$		$Pr(A_t=0,1)$	
000	1	0	.983	.017	.989	.011
001	1	0	1	0	.012	.988
010	.999	.001	.001	.999	.999	.001
011	1	0	0	1	.999	.001
100	0	1	1	0	1	0
101	0	1	1	0	0	1
110	0	1	0	1	1	0
111	.048	.952	0	1	.048	.952

For instance, the decoded sequence using the DBN model in Fig. 11 is obviously closer to the ground truth than that decoded using the static model.

It is clear why the DBN model performed better than the static one. Inherent temporal correlation of features was indeed exploited by the DBN. To see this, consider the entries of the transition probability table of the DBN, shown in Table 1.

These learned entries clearly indicate that strong correlation does exist between states at consecutive time instances.

The effect of temporal smoothing can be seen by comparing the decoded speaker state using static and dynamic BN models. Fig. 11 gives the results for a single test sequence. In comparing Fig. 11(b) to Fig. 11(c), it is clear that the DBN produces a much smoother estimate, without the rapid fluctuations in the static case (that produce the nearly solid black columns in the plot). At the same time, we observe that the temporal locality of the speaker events, espe-

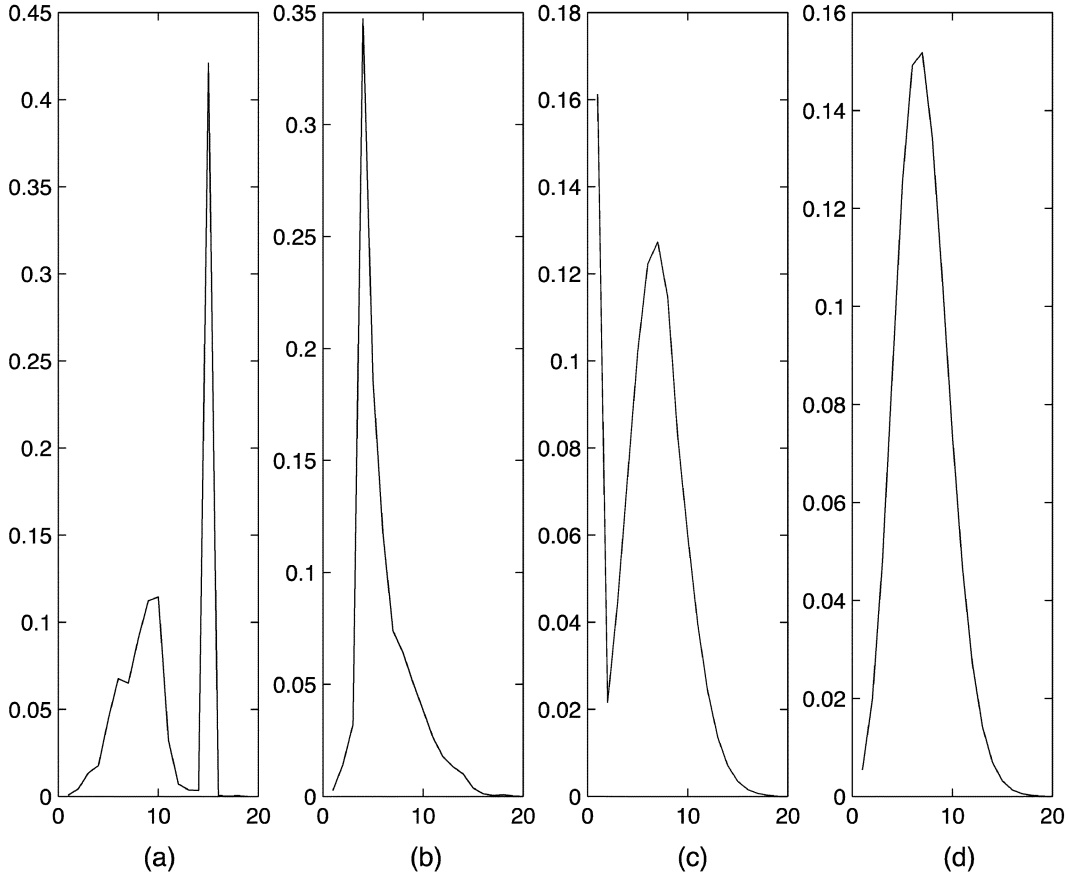


Fig. 12. Duration density plots for states of the DDDBN model. (a) No speaker, no frontal, no audio. (b) No speaker, no frontal, audio. (c) No speaker, frontal, no audio. (d) Speaker, frontal, audio states. The x axis gives the time instances, and the y axis gives the probability.

cially their duration, is misestimated by the DBN. Essentially what is happening is that the DBN is oversmoothing and extending the positive speaker state beyond the time interval in which it is appropriate.

C. Experiment Using Duration Density DBN

One approach to obtaining the right degree of smoothing in the DBN case is to apply explicit duration density estimation, resulting in a duration density DBN (DDDBN). DDDBNs are similar to explicit duration HMMs [25]. Duration densities for state durations of 1 up to 20 (time instances) were learned from the labeled data. Fig. 12 shows the learned CPTs. It is evident from these graphs that some of the duration distributions clearly differ from the exponential distribution imposed by the DBN model. The accuracy of the speaker detector improved slightly when DD modeling was used. An average accuracy of 83% was obtained. Fig. 11(d) shows an example of the decoded state sequence using the DDDBN model.

Nonetheless, improved performance of the DDDBN model is severely hampered by its complexity. The complexity of inference in DDDBNs increases exponentially with the duration of the states (compared to a DBN). This suggests that in practice it may be more profitable to retain the basic DBN architecture and modify the learning procedure to improve classification performance.

D. Experiment Using Static EFBN

As a baseline test, we also implemented an error-feedback version of the static BN (the EFBN). The EFBN was trained in the same fashion as its dynamic counterpart, using the Adaboost algorithm described in Section V. Unlike the EFDBN, there were no dynamic parameters to update in the EFBN algorithm—the only update was done on the measurement model parameter B_ϕ (see Section V). The performance of EFBN, as seen from Fig. 10, is on par with that of the DDDBN and the DBN. This is a clear example of the shortcomings of the ML estimation and the benefits of a discriminative learning approach, as discussed in Section V.

E. Experiment Using EFDBN

The final set of experiments concentrated on the newly designed EFDBN framework. We conducted two sets of experiments corresponding to discrete and continuous measurement variables. The learning algorithm described in Section V was used. For a training sequence, we used EFDBN to estimate the parameters which minimized the classification error. A leave-one-out cross validation resulted in an overall accuracy of 90.39%. Fig. 10 summarizes classification results on individual test sequences. We see that for all the sequences, an improvement of 5%–10% over the best DBN result is obtained.

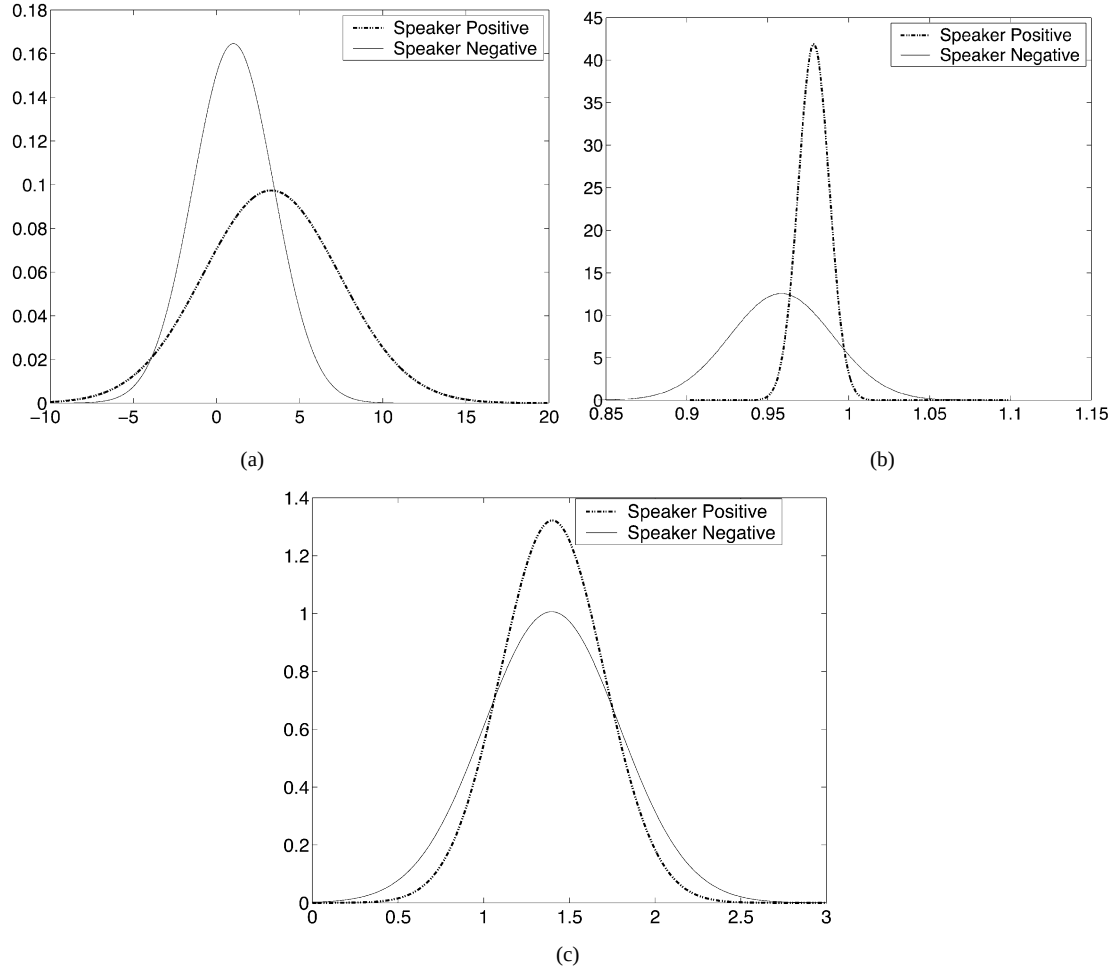


Fig. 13. Learned continuous sensor distributions: (a) silence, (b) skin texture, and (c) mouth motion.

One important issue deserves our comment: unless a classifier performs well on the training data, it cannot be expected to do a great job on the test data. During DBN training, we found the accuracy of classification on the training set of about 82%. This implies that one should not expect performance better than 82% on the test data (provided training data is representative). In the boosted scenarios, however, we were able to improve the performance on the training data to as much as 95%. Such performance can be attributed to the better-tuned representation power of the boosted ensemble models—a boosted model, as mentioned previously, is akin to a distribution mixture model. As expected, we also found a greatly improved performance on the test data.

The DBN model learned using the EFDBN framework was also applied to the prediction of hidden states. An overall accuracy of 88% was obtained. This indicates, together with the previously noted results, that EFDBN significantly improves the performance of simple DBN classifiers.

F. Continuous Sensory Measurements

The use of continuous valued sensor outputs allows the network to automatically learn optimal sensor models and, in turn, optimal decision thresholds. Here all continuous sensory outputs are modeled as conditional Gaussian distributions, as shown in Fig. 13. The learned distributions allow soft sensory decisions, which can be superior to discrete sensory outputs in noisy environments. For instance, it is well known that for a sensor whose distribution resembles that of our mouth motion sensor in Fig. 13(c), a single threshold decision results in high prediction error because both class densities have very similar means. However, because of their different variances the two classes may be distinguished using maximum *a priori* estimation and explicit distribution models. Note again that all of the continuous representations point to the ambiguity of the sensors—all sensory measurements are characterized by highly overlapping class distributions.

Fig. 14 displays results of using different integration frameworks with soft sensory measurement and their comparison to the decisions made by models with discrete sensory decision.

In all cases, except for sequence 5 and partially sequence 1, standard multimodal integration topologies (BN and DBN) benefited significantly and expectedly from “soft decision” sensors. Improvements of up to 16% were observed on sequence 4 and the basic BN model. Interestingly, the discriminative EFDBN model exhibited relatively minor performance improvement with the introduction of continuous sensors. This, indeed, could be attributed to the better (and possibly sufficiently good) representation of the

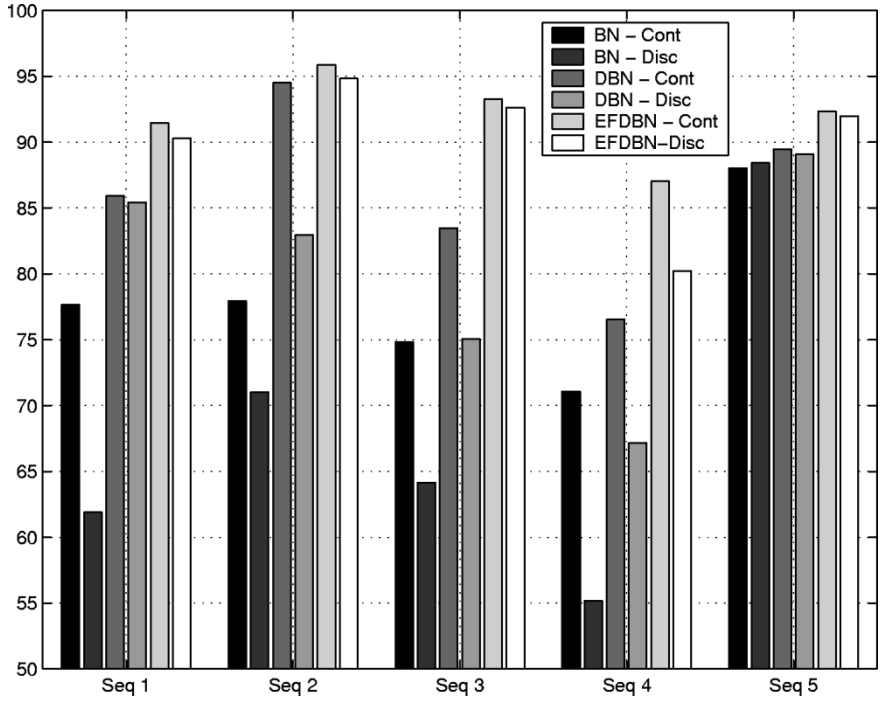


Fig. 14. Comparison of speaker prediction accuracy (%) obtained using static BN, DBN, and EFDBN. 'Cont' and 'Disc' denote the models with continuous and discrete sensory inputs, respectively.

true underlying data distribution by the ensemble model. Similarly surprising was the almost uniformly good performance, around 90% accuracy, of all integration models on sequence 5. Again, as noted in the previous sections, sequence 5 contained somewhat "cleaner" data than the other four sequences.

VIII. Conclusion

We have demonstrated the utility and practicality of using boosted learning to improve the performance of BN classifiers. Our learning framework makes it possible to retain all of the positive aspects of BNs for classification tasks, without sacrificing classification accuracy. These positive aspects include interpretability, the use of sampling and sensitivity analysis to analyze learned representations, and great facility in handling time series data. Our boosting formulation converts the standard parameter learning procedures for BN and DBN models into supervised procedures. This requires only minimal modification of the core parameter learner and adds only a constant factor to the computational cost of the training and testing phases.

We present a range of experimental results which validate the benefit of our boosting framework. The context for our experiments is a novel application of multimodal sensing to a Smart Kiosk user interface. Audiovisual sensing is used to detect when a user is speaking to the kiosk. This capability can gate the application of a command-and-control speech interface and support new forms of speech-based interaction. We present experimental results for a speech-based interface to a multiperson blackjack game, called the Genie Casino.

Acknowledgment

This research was conducted at the Cambridge Research Laboratory, Compaq Computer Corporation, Cambridge, MA, and at the University of Illinois, Urbana-Champaign. The authors would like to thank A. Christian and B. Avery of Cambridge Research Laboratory, the creators of the Genie Casino blackjack game, for their help with the experiments.

References

[1] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "Estimating hidden Markov model parameters so as to maximize speech recognition accuracy," *IEEE Trans. Speech Audio Processing*, vol. 11, pp. 77–82, 1993.

[2] M. Beal, N. Jojic, and H. Attias, "A self-calibrating algorithm for speaker tracking based on audio-visual statistical models," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2002, pp. 1997–2000.

[3] A. D. Christian and B. L. Avery, "Digital Smart Kiosk project," in *Proc. ACM S1GCH1 '98*, 1998, pp. 155–162.

[4] R. Cutler and L. Davis, "Look who's talking: Speaker detection using video and audio correlation," *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, pp. 1589–1592, 2000.

[5] T. G. Dietterich, "Machine-learning research," *AI Mag.*, vol. 18, pp. 97–136, 1997.

[6] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussions)," *Ann. Stat.*, vol. 28, pp. 337–407, 2000.

[7] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, pp. 131–163, 1997.

[8] A. Garg, V. Pavlovic, J. M. Rehg, and T. S. Huang, "Audio-visual speaker detection using dynamic Bayesian networks," in *Proc. of 4th Int. Conf. Automatic Face and Gesture Recognition*, 2000, pp. 374–471.

[9] A. Garg, "Learning dynamic Bayesian networks for multimodal speaker detection," master's thesis, Dept. Elect. Eng., Univ. Illinois, Urbana-Champaign, 2000.

[10] D. Heckerman, "A tutorial on learning with Bayesian networks," Microsoft Research, Tech. Rep. MSR-TR-95-06, 1995.
[11] J. W. Fisher III, T. Darrell, W. T. Freeman, and P. Viola, "Learning joint statistical models for audio-visual fusion and segregation," presented at the Advances Neural Information Processing Systems, Denver, CO, 2000.
[12] T. Jabara and A. Pentland, "On reversing Jensen's inequality," *Neural Inf. Process. Syst.*, vol. 13, pp. 231–237, Dec. 2000.
[13] F. V. Jensen, *An Introduction to Bayesian Networks*. New York: Springer-Verlag, 1996.
[14] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *Int. J. Comput. Vis.*, vol. 46, no. 1, pp. 81–96, Jan 2002.
[15] M. I. Jordan and R. A. Jacobs, "Hierarchial mixture of experts and the em algorithm," *Neural Comput.*, vol. 6, pp. 181–214, 1994.
[16] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Signal Processing*, vol. 40, pp. 3043–3054, Dec. 1992.
[17] D. Koller and A. Pfeffer, "Object-oriented Bayesian networks," in *Proc. 13th Conf. Uncertainty AI*, 1997, pp. 302–313.
[18] A. Krieger, C. Lung, and A. Wyner, "Boosting noisy data," in *Proc. Int. Conf. Machine Learning*, 2001, pp. 274–281.
[19] P. J. Moreno, B. Logan, and B. Raj, "A Boosting approach for confidence scoring," HP Labs, Tech. Rep. CRL 2001/8, 2001.
[20] N. Oliver, A. P. Pentland, and F. Bérard, "Lafter: Lips and face real time tracker," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 123–129.
[21] D. Optiz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, 1999.
[22] V. Pavlović, A. Garg, J. M. Rehg, and T. Huang, "Multimodal speaker detection using error feedback dynamic Bayesian networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 34–41.
[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1998.
[24] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, pp. 257–286, Feb. 1989.
[25] L. R. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[26] J. M. Rehg, M. Loughlin, and K. Waters, "Vision for a Smart Kiosk," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 690–696, 1997.
[27] J. M. Rehg, K. P. Murphy, and P. W. Fieguth, "Vision-based speaker detection using Bayesian networks," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 110–116, 1999.
[28] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 203–208, 1996.
[29] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, pp. 297–336, 1999.
[30] R. E. Schapire, "A brief introduction to boosting," presented at the Int. Joint Conf. Artificial Intelligence, Stockholm, Sweden, 1999.
[31] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Stat.*, vol. 26, no. 5, pp. 1651–1686, Oct. 1998.
[32] H. Schwenk, "Using boosting to improve a hybrid hmm/neural networ speech recognizer," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 1999, pp. 1009–1012.
[33] O. Siohan, A. E. Rosenberg, and S. Parthasarathy, "Speaker identification using minimum classification error training," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 1, 1998, pp. 109–112.
[34] J. Vermaak, M. Gangnet, A. Blake, and P. Pérez, "Sequential Monte Carlo fusion of sound and vision for speaker tracking," in Intl. Conf. Computer Vision, Vancouver, BC, Canada, 2001.
[35] J. Yang and A. Waibel, "A real-time face tracker," in *Proc. 3rd Workshop Applications Computer Vision*, 1996, pp. 142–147.

Ashutosh Garg (Member, IEEE) received the B.S. degree in electrical engineering from the Indian Institute of Technology, Delhi, in 1993 and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana-Champaign, in 2000 and 2002, respectively.

He is currently a Research Staff Member with IBM Almaden Research Center, San Jose, CA, where he is working in the area of business intelligence from structured and unstructured data. He has published numerous papers in journals and leading international conferences and is a author of three patent applications. His research interests include machine learning, multimedia analysis, and text analytics.

Dr. Garg has received the Robert T. Chien Award from the Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, for demonstrated excellence in research.

Vladimir Pavlović (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1999.

From 1999 until 2001, he was a Member of the research staff at the Cambridge Research Laboratory, Cambridge, MA. He is currently an Assistant Professor in the Computer Science Department, Rutgers University, Piscataway, NJ, and an Adjunct Assistant Professor in the Bioinformatics Program, Boston University, Boston, MA. His research interests include statistical modeling of time-series, statistical computer vision, machine learning, and bioinformatics.

James M. Rehg received the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, in 1995.

From 1996 to 2001, he led the computer vision research group at the Cambridge Research Laboratory, Cambridge, MA. In 2001, he joined the faculty of the College of Computing, Georgia Institute of Technology, Atlanta, GA, where he is currently an Associate Professor. His research interests include computer vision, machine learning, human–computer interaction, computer graphics, and distributed computing.