Appears in *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages II:34-41, Hilton Head, SC, June 13-15, 2000.

Multimodal Speaker Detection using Error Feedback Dynamic Bayesian Networks

Vladimir Pavlović[1], Ashutosh Garg[2], James M. Rehg[1], and Thomas S. Huang[2]

[1]Compaq Computer Corporation
Cambridge Research Lab
Cambridge, MA 02139
{vladimir,rehg}@crl.dec.com

[2]Beckman Institute and Coordinated Science Lab
University of Illinois
Urbana, IL 61801
{ashutosh,huang}@ifp.uiuc.edu

Abstract

Design and development of novel human-computer interfaces poses a challenging problem: actions and intentions of users have to be inferred from sequences of noisy and ambiguous multi-sensory data such as video and sound. Temporal fusion of multiple sensors has been efficiently formulated using dynamic Bayesian networks (DBNs) which allow the power of statistical inference and learning to be combined with contextual knowledge of the problem. Unfortunately, simple learning methods can cause such appealing models to fail when the data exhibits complex behavior. We formulate a learning framework for DBNs based on error-feedback *and statistical boosting theory. We apply this framework to the problem of audio/visual speaker detection in an interactive kiosk environment using "off-the-shelf" visual and audio sensors (face, skin, texture, mouth motion, and silence detectors). Detection results obtained in this setup demonstrate superiority of our learning framework over that of the classical ML learning in DBNs.*

1. Introduction

Human-centered user-interfaces based on vision and speech present challenging sensing problems in which multiple sources of information must be combined to infer the user's actions and intentions. Statistical modeling techniques therefore play a critical role in system design. Dynamic Bayesian network (DBN) models are an attractive choice, as they combine an intuitive graphical representation with efficient algorithms for inference and learning. DBNs are a class of graphical probabilistic models which encode dependencies among sets of random variables evolving in time. Examples of DBNs include Kalman filters and HMMs. Previous work has demonstrated the power of these models in fusing video and audio cues with contextual information and expert knowledge [6, 8, 7, 3].

Speaker detection is a particularly interesting example of a multi-modal sensing task which can serve as a testbed for DBN research. Detecting when users are speaking is an important component of an open mike speech-based user-interface. The need to handle multiple people in the presence of background noise means that both audio- and video-based sensing can provide useful information. Figure 6 gives an example of a DBN model for speaker detection. We are interested in network models that combine "off-the-shelf" vision and speech sensing with contextual cues such as the state of the interaction. Previous work has demonstrated promising results for speaker detection using both static Bayesian networks [11] (BNs) and, more recently, DBNs [6].

Learning the parameters of a DBN model is a key step in the development of an effective system. The complexity of these models and the number of free parameters make hand-tuning impractical. Maximum likelihood (ML) learning is the most common approach, in which model parameters are adjusted to achieve the best fit to a set of training data. Unfortunately there is no guarantee that a model which fits its training data well will make a good classifier. However, a recent learning technique known as *boosting* makes it possible to improve the accuracy of a weak classifier through error-feedback and the optimal combination of multiple classifiers [13]. Boosting algorithms develop a series of classifiers that concentrate on the errors made by their predecessors, thereby improving performance.

This paper describes a novel learning algorithm for DBNs that uses boosting to improve recognition accuracy. We refer to this new model as an *error feedback DBN* or EFDBN. It combines boosting with ML learning for estimating the model parameters. We demonstrate the utility of this new learning approach in the context of speaker detection. Our experiments show that the EFDBN is superior to conventional ML-based DBN models, and that both are superior to static BNs. On a test set of five sequences we

achieve an accuracy of 90%. These promising results suggest the general applicability of EFDBN to other problems in which BN and DBN models are in use.

The rest of the paper is organized as follows. We begin by introducing the problem of multi-sensor fusion for speaker detection in Section 2, followed by a brief review of the previous static and dynamic BN approaches. Section 3 formally addresses the classical ML learning in DBNs. In Section 5 we define our EFDBN learning framework from the prospective of boosting and discuss its relation to other similar schemes. Section 6 describes the experiments on speaker detection using the three frameworks (static BN, DBN, and EFDBN.) Lastly, we provide some final discussion of the framework and the results, followed by the future research directions.

2. Speaker Detection

Speaker detection is an important component of open-mike speech-based user-interface. For any interface which relies on speech for communication, an estimate of the persons state (whether he/she is or isn't a speaker) is important for its reliable functioning. We argue that for a person to be an active user (speaker), he must be expected to speak, facing the system and actually speaking. Visual cues can be useful in deciding whether the person is facing the system and whether he is moving his lips. However, they are not capable on their own to distinguish an active user from an active listener (listener may be smiling or nodding). Audio cues, on the other hand, can detect the presence of relevant audio in the environment. Unfortunately, simple audio cues are not sufficient to discriminate a user in front of the system speaking to the system from the same user speaking to another individual. Finally, contextual information describing the "state of the world" also has bearing on when a user is actively speaking. For instance, in certain contexts the user may not be expected to speak at all. Hence, audio and visual cues as well as the context need to be used jointly to infer the active speaker.

The Smart Kiosk [10, 4] developed at Compaq's Cambridge Research Lab (CRL) provides an interface which allows the user to interact with the system using spoken commands. The public, multi-user nature of the kiosk application domain makes it ideal as an experimental setup for speaker detection task. The kiosk (see Figure 1) has a camera mounted on the top that provides visual feedback. A microphone is used to acquire speech input from the user. This setup forms an ideal testbed for our problem.

We have analyzed the problem of speaker detection in a specific scenario of the Genie Casino Kiosk. This version of kiosk simulates a multiplayer blackjack game (see Figure 7 for a screen capture.) The user uses a set of spoken commands to interact with the dealer (kiosk) and play the game.

Figure 1. The CRL Smart Kiosk

Audio and visual information can be obtained directly from the two kiosk sensors. We use a set of five "off-the-shelf" visual and audio sensors: the CMU face detector [12], a Gaussian skin color detector [15], a face texture detector, a mouth motion detector, and an audio silence detector. A detailed description of these detectors can be found in [11]. Contextual sensor provides the state of the environment which may help in inferring the state of the user. Contextual information can tell whether the user is expected to speak or not.

2.1. Bayesian Networks for speaker detection

The speaker detection problem represents a challenging ground for testing the representational power of DBN models and more specifically the EFDBN algorithm in a complex multi-sensor fusion task. Different types of sensors need to be seamlessly integrated in 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, data-driven aspect comes into play with data-driven parameter learning.

The graph in Figure 2 shows the vision network for this task. This network takes the binary output of the sensors (skin color detector, face detector and texture detector) and outputs the query variables corresponding to visibility and the frontal information of the user.

The silence detector and the mouth motion detector are used to infer whether the user is talking. The audio network

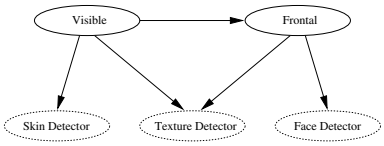

Figure 2. Vision Network

selected for this task is shown in Figure 3. It takes the input from the sensors and outputs the probability that the audio present in the environment corresponds to the user.

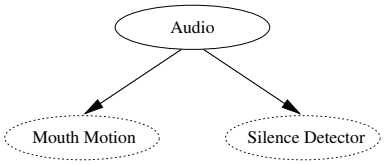

Figure 3. Audio network for speaker detection.

Once constructed, the audio and visual networks are fused to obtain the integrated audio–visual network. At this stage one would also like to incorporate any information the environment may play in deciding the user's state. The contextual information (state of the blackjack game), together with the visual and audio subnetworks is now fused into a single net through the virtue of the speaker node, as shown in Figure 4. The chosen network topology represents our knowledge that both audio, visual, as well as contextual conditions need to be met for the decision on the presence of the speaker to be made.

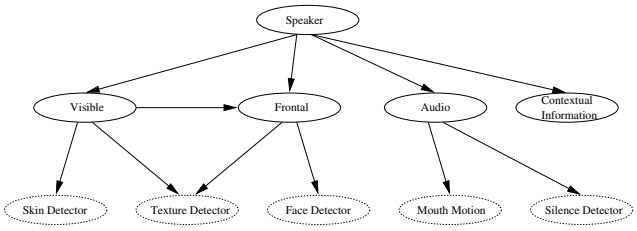

Figure 4. Integrated audio-visual network.

The final step in designing the topology of the speaker detection network involves its temporal aspect. Measurement information from several consecutive time steps can be fused to make a better informed decision. This expert knowledge becomes a part of the speaker detection network once the temporal dependency shown in Figure 5 is imposed. The presence of all possible arcs among the three nodes stems from our lack of exact knowledge about these temporal dependencies, i.e., we allow for all dependencies to be present and later on determined by the data.

Incorporating all of the above elements into a single structure lead to the DBN shown in Figure 6. Here the nodes

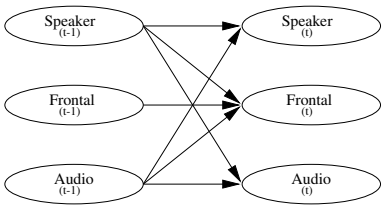

Figure 5. Temporal dependencies between the speaker, audio, and frontal nodes at two consecutive time instances.

shown in dotted lines are the direct observation nodes while the ones in solid are the unobserved nodes. The speaker node is the final speaker detection query node.

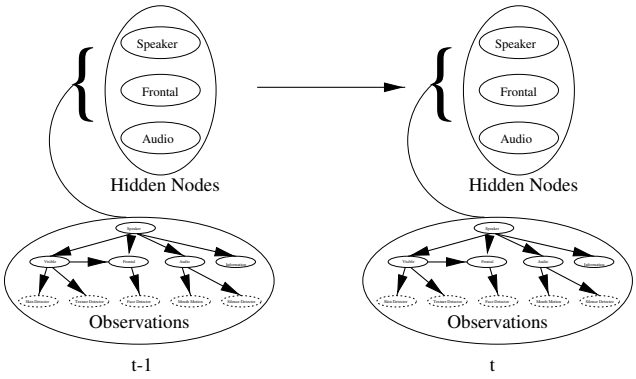

Figure 6. Two time slices of the dynamic Bayesian network for speaker detection.

It has been shown in [11] and [6] how both the speaker detection models based on the static BN, similar to the one in Figure 4, and the DBN in Figure 6 can be learned from data using standard ML learning techniques and then effectively utilized to fuse multi-sensory information. The DBN framework has been show in [6] to outperform the static one mainly due to the existence of temporal links but also because of the presence of contextual information. A significant improvement of 15% in speaker detection error rate was reported. Further improvements occured when higher order temporal dependencies were introduced through duration density DBNs.

3. ML Learning in Dynamic Bayesian Networks

Dynamic Bayesian networks are a class of Bayesian networks specifically tailored to model temporal consistency present in some data sets. In addition to describing dependencies among different static variables DBNs [5] describe probabilistic dependencies among variables at differ-

ent time instances. In general, a DBN has a specific structure shown in an example in Figure 6. A set of random variables at each time instance t is represented as a static BN. Out of all the variables in this set temporal dependency is imposed on some. Namely, distribution of some variable $s_{i,t}$ at time t depends on a variable at time $t-1$, $s_{j,t-1}$ through some conditional distribution Pr $(s_{i,t}|s_{j,t-1})$. An example of this structure is depicted in Figure 6. Finally, some variables at each time slice are considered to be observable (sensor measurements) and are usually denoted by y_t. The rest of the variables can but need not be observed. Probability distribution among all variables in a DBN can in general be written as Pr $(y_1, \ldots, y_T, s_1, \ldots, s_T) =$ Pr (s_1)Pr $(y_1|s_1) \prod_{t=2}^{T}$ Pr $(s_t|s_{t-1})$Pr $(y_t|s_t)$. Each of the Pr terms can be either a table of probabilities of some parametric pdf. In both cases, they yield a set of model parameters, which we denote by Θ. In general, Θ consists of three types of parameters: transition probability parameters A, static BN parameters B, and initial state distribution π.

Inference in DBNs is concerned with finding the distributions (i.e., estimates $\hat{s_t}$) of unobserved variables s_t given the measurements $y_1, \ldots, y_T$. Thanks to its constrained topology efficient algorithms such as the forward-backward propagation [2] can be employed for this task.

ML learning in DBNs is a special case of ML learning in general BNs. The goal is to maximize the likelihood of observed variables by varying the model's parameters. Given the DBN pdf it is easy to formulate the ML learning as

$$\Theta^* = \arg\max_{Theta} \Pr(\text{observed variables}|\Theta). \qquad (1)$$

The optimization usually has a closed-form solution when all the variables in the DBN are visible and the optimal values of three parameters A, B, and π are independent. If some of the variables (x) are hidden, the closed-form solution is usually replaced by an iterative procedure known as the expectation-maximization or EM:

Get initial guess of Θ_0;
do
  Infer hidden variables $\hat{s}_t$ from measurements
    $y_1, \ldots, y_T$ using model Θ_k;
  $\Theta^*_{k+1} = \arg\max_{Theta} \Pr(y_1, \ldots, y_t, \hat{s}_1, \ldots, \hat{s}_T|\Theta)$.
until (convergence)

Because three types of parameters are present, A, B, and π, the iterations can be formulated such that all parameters are updated at one time or only some of them are updated while the others are held fixed. It is important to note that in this case the optimal values of parameters can depend on each other.

4. Classification Error and Boosting

ML estimators have an undeniable appeal. The arguments in favor of ML estimation are based on the assumption that the form of the underline distribution is known, and that only the value of the parameters characterizing the distribution is unknown. However, maximizing the likelihood does not necessarily lead to minimum classification error, an important criterion in problems such as multi-sensor speaker detection.

Recently, Schapire et al. [13] have proposed method called boosting aimed at improving the performance of any weak classifier. In particular, they have derived an algorithm called Adaboost that "boosts" the classification on a set of data points by linearly combining a number of weak classifiers, each of which is trained to correct "mistakes" of the previous one.

More formally, consider a binary classification problem with data given by $S = \{(s_1, y_1), ..., (s_m, y_m)\}$. Here y_i is a feature vector and s_i is the desired label (or ground truth). The goal of the learning algorithm is to find a hypothesis (classifier) $h : Y \to S$ that minimizes misclassification. In a binary classification scenario, $s \in \{+1, -1\}$, Adaboost can be described as

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

- Train hypothesis h_k using data D with distribution $P_D^{(k)}$.
- Choose $\alpha_k = \frac{1}{2} \ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$
  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
$H(y) = sign\left(\sum_{k=1}^{K} \alpha_k h_k(y)\right)$

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

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

Hence, if the weak hypotheses are slightly better than the chance, the training error decreases exponentially fast. Additional bounds on the generalization error can also be derived [13]. It has also been shown empirically that Adaboost

has a good generalization property, unless the number of hypothesis becomes too large. Extensions of Adaboost to multilabel and soft classification problems have also been reported.

5. Error Feedback DBNs

Consider the training data $D = \{(s_1, y_1), ..., (s_m, y_m)\}$, and the DBN shown in Figure 6. The modified goal of DBN learning can now be described as: given data D obtain DBN model $\Theta = (A, B, \pi)$, which minimizes the probability of classification error in s on dataset D. EFDBN algorithm for this setting can now be formulated as follows.

Given: $D\{(s_1, y_1), ..., (s_T, y_T)\}$;
where y_t is an observation vector and s_t is the corresponding label of the hidden state

Initialize $P_D^{(1)}(i) = 1/m$;
For $k = 1, ..., K$

- Train static BN[1] with s_t as the root node to obtain B_k.
  Use $P_D^{(k)}$ as the weight over the training samples.
- Use the DBN learning algorithm to obtain the the transition probability matrix A for fixed B_k.
- Use the learned DBN, $\Theta = (A, B_t, \pi)$ to decode the hidden state sequence $(\hat{s}_1, ...\hat{s}_T)$ given $(y_1, ..., x_T)$ as the input:
  $\hat{s}_t = argmax_i P(s_t = i | y_1, x_2, ...x_T)$
- Choose $\alpha_k = \frac{1}{2} \ln \left(\frac{1-\epsilon_k}{\epsilon_k} \right)$
  where $\epsilon_k = Pr_{t \sim P_D^{(k)}}[\hat{s}_t \neq s_t]$
- Update:
  if $\hat{s}_k = s_k$ then
  $P_D^{(k+1)}(t) = \frac{P_D^{(k)}(t) \exp(-\alpha_k)}{Z_k}$
  else
  $P_D^{(k+1)}(t) = \frac{P_D^{(k)}(t) \exp(\alpha_k)}{Z_k}$
  where Z_k is the normalization factor.

The final HMM model is $\lambda = (A, B, \pi)$
where $B = \frac{\sum_{k=1}^{K} \alpha_k B_k}{\sum_{k=1}^{K} \alpha_k}$

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 A, for which all the sample are considered to be equally probable. Once DBN is trained, 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 uses the weighted sum of individual observation probability matrices. The weight of the individual probability matrix is a function of the expected error made by that model. This DBN is now used for classification. Since B_k gives the confidence in a certain state and not the binary decision, we need to modify the way we are measuring the error:

$$\epsilon_k = E_{t \sim P_D^{(k)}}[h_t s_t], \tag{3}$$

where $h_t = (2 * Pr(\hat{s}_t | y_1, ..., y_T) - 1)$ (i.e., $h_t \in [-1, +1]$.) It has been shown in [13] that the bound on training error of Eqn 2 still holds. This algorithm can be extended easily to the case when s_i takes multiple values by using the multiple class version of Adaboost [13].

Algorithms similar in flavor to ours have appeared in recent literature. In [1], the authors suggested the use of corrective training for improving the performance of hidden Markov models (simple DBNs) in a speech recognition framework. While improved performance compared to the standard HMM classification was reported, certain convergence issues remained at stake. The use of Adaboost to train the hybrid HMM/neural network speech recognizer was recently reported in [14]. The Adaboost was utilized to enhance the performance of the neural network measurement model, hence resulting in better overall recognition performance.

6. Experiments and Results

We conducted three experiments using a common data set. The data set comprised of five sequences of a user playing the blackjack game in the Genie Casino Kiosk setup. The exerperimental setup is depicted in Figure 7. The sequences were of varying duration (from 2000 samples to 3000 samples) totaling to 12500 frames. Figure 8 shows some of the recorded frames from the video sequence. Each sequence included audio and video tracks recorder through a camcorder along with frequency encoded contextual in-

[1]During training all the nodes of the BN are considered to be observable. If that is not the case, EM algorithm needs to be used for learning the BN with the hidden nodes.

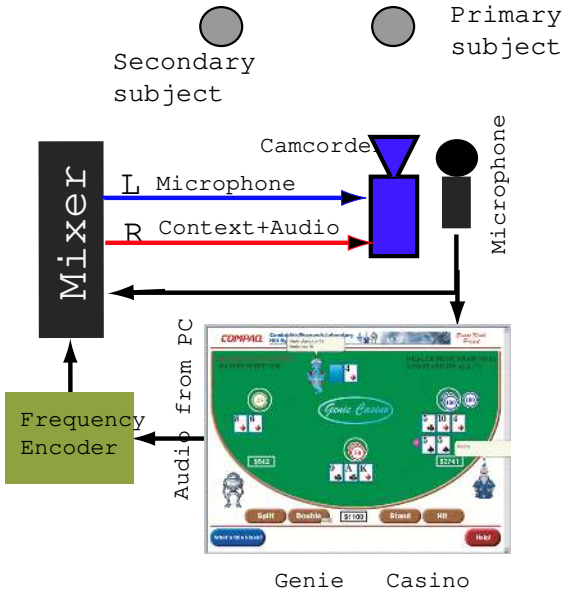

Figure 7. Data collection set-up for Genie Casino kiosk.

formation (see Figure 7.) The visual and audio sensors were then applied to audio and video streams. Because some of the sensors provide continuous estimates of their respective functions, decision thresholds were determined for each sensor that yield binary sensor states (e.g., silence v.s. no silence.) These discretized states were then used as input for the DBN model. Examples of individual sensor decisions (e.g., frontal v.s. non frontal, silence v.s. non silence, etc.) are shown in Figure 9. Abundance of noise and ambiguity in these sensory outputs clearly justifies the need for intelligent yet data-driven sensor fusion.

Figure 8. Three frames from a test video sequence.

6.1. Experiment Using Static BN

The first experiment was done using the static BN of Figure 4 to form the baseline for comparison with the dynamic model. In this experiment all samples of each sequence was considered to be independent of any other sample. Part of the whole data set was considered as the training data and 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.

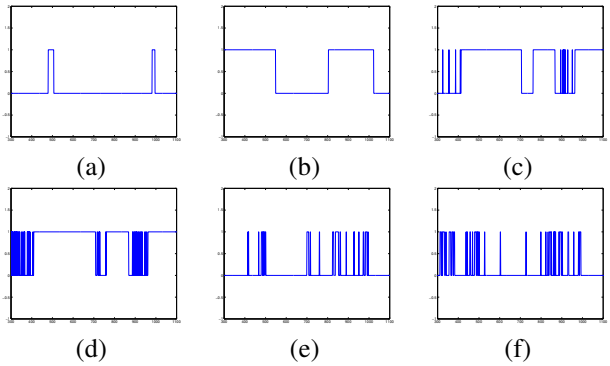

Figure 9. Figure (a) shows the ground truth for the speaker state. 1 means that there is a speaker and 0 means an absence. x axis gives the frame no. in the sequence. (b) gives the contextual information. 1 means, its users turn to play where as 0 means the computer is going to play. (c),(d),(e),(f) are the output of texture, face, mouth motion and silence detector respectively.

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. Cross validation was done by choosing different training and test data. An average accuracy of 65% is obtained (see Figure 10 for results on individual sequences.) The accuracy obtained is low. The sensor data (as shown in Figure 9) is noisy and it is hard to infer the speaker without making substantial errors. Figure 11(a) shows the ground truth sequence for the state of the speaker and (b) shows the decoded sequence using static

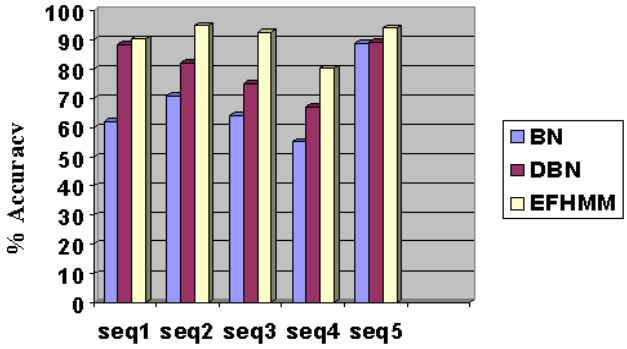

Figure 10. A comparison between the results obtained using static BN, DBN, EFDBN

BN. On the other hand, temporal consistency in the query state (speaker ground truth) indicates that a model should be built that exploits this fact.

6.2. Experiment Using DBN

Second 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 3 was employed to learn the dynamic transitional probabilities among frontal, speaker, and audio states. During testing phase a temporal sequence of sensor values was presented to the model and Viterbi decoding (c.f. [9]) 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 15% over the static BN model. An indicative of this can be seen in actual decoded sequences. For instance, decoded sequence using the DBN model in Figure 11 is obviously closer to the ground truth than the one decoded using the static model. The improved performance by the use of DBN stems from the inherent temporal correlation present between the features.

6.3. Experiment using EFDBN

Our final experiment employed the newly designed EFDBN framework. The learning algorithm described in Section 5 was used. For a training sequence, we used EFDBN to estimate the parameters which minimized the classification error. A leave-one-out crossvalidation resulted in the overall accuracy of 90.39%. Figure 10 summarizes classification results on individual sequences. We see that for all the sequences, an improvement of $5 - 10\%$ over the best DBN result is obtained.

One additional 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 anything better than 82% on the test data (provided training data is a representative of the test data). Fortunately, this is where boosting comes into play. It takes a weak classifier (which showed poor performance on the training data) and enhances its performance. In our case, by doing boosting, we were able to improve the performance on the training data to as much as 93%. 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.

7. Discussions and Conclusions

We have presented a general purpose error-feedback learning framework for DBNs. The results obtained for the difficult problem of speaker detection where a number of noisy sensory outputs need to be fused indicate the utility of this algorithm. Significant improvements in classification accuracy over a simple DBN model were achieved without sacrificing of complexity of the learning algorithm. We have also demonstrated a general purpose approach to solving man-machine interaction tasks in which DBNs are used to fuse the outputs of simple audio and visual sensors while exploiting their temporal correlation.

In future work, we will focus on extending the boosting to encompass the transition parameters. Our initial experiments indicate that convex combination of transition matrices obtained in a manner similar to the one used for observation matrix B does not yield significant improvements in performance. We will also like to point that the bound on the error (given in Eqn 2) may no longer hold because of the temporal dependence between the data[2] . Our current research focuses on obtaining the bounds for this case.

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. In *IEEE Transcations on speech and audio processing*, pages 77–82, 1993.

[2] X. Boyen, N. Firedman, and D. Koller. Discovering the hidden structure of complex dynamic systems. In *Proc. of the Conf. on Uncertainty in AI*, pages 91–100, 1999.

[3] 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.

[4] A. D. Christian and B. L. Avery. Digital smart kiosk project. In *ACM SIGCHI '98*, pages 155–162, Los Angeles, CA, April 18–23 1998.

[5] T. Dean and K. Kanazawa. A model for reasoning about persistance and causation. *Computational Intelligence*, 5(3), 1989.

[6] 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.

[7] S. Intille and A. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks. Technical Report 454, MIT Media Lab, 1998.

[8] V. Pavlovic, B. Frey, and T. S. Huang. Time-series classification using mixed-state dynamic Bayesian networks. In *Computer Vision and Pattern Recognition*, June 1999.

[2] Because of the temporal dependence, errors may propagate from one time to the other.

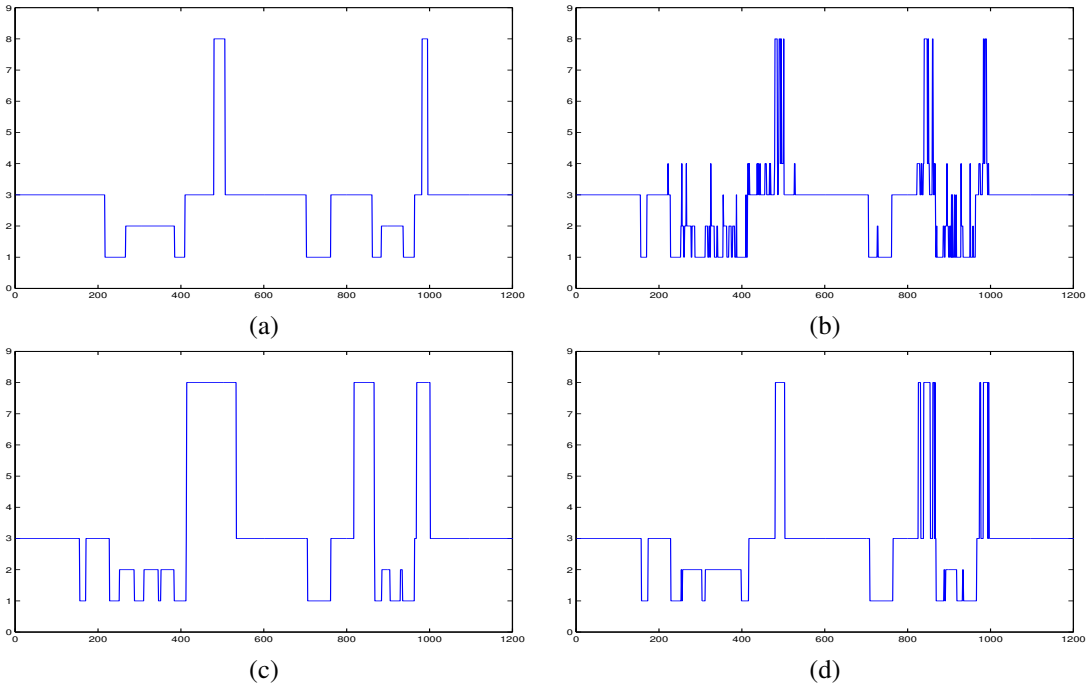

Figure 11. Figure (a) shows the true state sequence. (b),(c),(d) are the decoded state sequences by static BN, DBN, EFDBN respectively. (state 1 - no speaker, no frontal, no audio; state 2 - no speaker, no frontal, audio; state 3 - no speaker, frontal, no audio; state 8 - speaker, frontal, audio)

[9] L. R. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[10] 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.

[11] 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.

[12] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[13] R. E. Schapire and Y. Singer. Improved boosting algorithms using cofidence rated predictions. *Machine Learning*, 1999. to appear.

[14] H. Schwenk. Using boosting to improve a hybrid hmm/neural network speech recognizer. In *ICASSP*, pages 1009–1012, 1999.

[15] J. Yang and A. Waibel. A real-time face tracker. In *Proc. of 3rd Workshop on Appl. of Comp. Vision*, pages 142–147, Sarasota, FL, 1996.