Bayesian Networks as ensemble of Classifiers*

Ashutosh Garg
Beckman Institute
University of Illinois
Urbana, IL 61801

Vladimir Pavlović
Boston University
Boston, MA 02215

Thomas S. Huang
Beckman Institute
University of Illinois
Urbana, IL 61801

Abstract

Classification of real-world data poses a number of challenging problems. Mismatch between classifier models and true data distributions on one hand and the use of approximate inference methods on the other hand all contribute to inaccurate classification. Recent work on boosting by Schapire et al. and additive probabilistic models by Hastie et al. have shown that improved classification can be achieved by linearly combining a number of simple classifiers. Building upon this spirit, we present a Bayesian network-based framework for mixing multiple classifiers. We also analyze the bound on the generalization error for this combined classifier. We give results on some standard datasets and demonstrate its usefulness in a real-world task of multimodal speaker detection where we improve upon performance of a more complex Bayesian network model. Improved results indicate the significant potential of the Bayesian network of classifiers approach.

1 Introduction

Combining the predictions of multiple classifiers has been shown to improve classification error rate as compared to the error rate obtained by learning a single model of the data. Recently, many researchers [1, 10, 9, 3] have demonstrated that using classifier ensembles leads to improved performance for many difficult generalization problems.

In this paper, we propose a Bayesian networks to efficiently combine simple classifiers. We show that if a Bayesian network is used to combine the classifiers, the classification performance of the combined classifier is going to be at least as well as the best classifier (on the training data) in the set, under fairly weak conditions and in practice this combined classifier indeed performs much better than any individual classifier in the set. It allows the classifiers to be trained independently of each other, and combined based on their *joint* performance on the training data. It turns out that the bounds given for decision tree [5] can be extended to bound the generalization performance of the combined classifier. We present a novel algorithm that forms a decision tree of classifiers by ordering the classifiers in order of their performance on the training data. We refer to a decision tree obtained in this way as the pruned decision tree. Because of the small number of leaf nodes in the decision tree obtained as such, we show that much smaller bound on generalization error is obtained.

We give results on a number of standard datasets from UCI ML repository and DELVE database and compare the performance with Adaboost [9] (a popular technique to boost the classification performance.) We also analyze the performance of this technique in solving the problem of real-world speaker detection. We show that the Bayesian network of classifiers improves classification performance significantly using linear perceptron and Bayesian networks as component classifiers.

2 Bayesian Network of Classifiers

Let us consider two popular techniques of combining the classifiers. 1) Supra Bayesian approach and 2) Stacking Algorithm. The principle underlying the Supra Bayesian approach is that from the viewpoint of the decision maker (the final classification), the opinions expressed by experts (base classifier) are data. Consequently, a decision maker combines the probability distributions provided by the individual experts via Bayes' rule. Let 's' be the quantity of interest (the output of the classifier), $P_i = p_i(s|H_i)$ denote the expert i's probability distribution for 's' when its knowledge is H_i, then the decision maker would base its decision on $p(s|H, P_1, ..., P_m)$ where H is the knowledge decision maker has about the event 's'.

Wolpert's stacking work as follows: suppose we have L different learning algorithms $A_1, ..., A_L$ and a set S of training examples $\{(x_1, s_1), ..., (x_m, s_m)\}$, where $x_i \in X$ and $s_i \in \{-1, 1\}$. One applies each of the training algorithms to the training data to produce classifiers $h_1, ...h_L$ s.t. $h_i : X \rightarrow \{-1, 1\}$. The goal of the stacking is to learn a good combining classifier h^* such that the final classification will be computed by $h^*(h_1(\mathbf{x}), ..., h_L(\mathbf{x}))$.

Motivated by these two algorithms, we propose a Bayesian network (BN) based approach to combine the classifiers. The idea is to use the BN structure shown in Figure 1 (a) to combine the classifiers. The BN can be thought

*This work was supported in part by the following grants: NSF EIA 99-75019, NSF IIS 00-85980, and NSF KDI.

of as classifier h^* in Wolpert's stacking algorithm or it can be thought of as a decision maker making its decision based on $p(s|h_1,...,h_m)$. In [6], it is shown that an ensemble can be more accurate only if the individual classifiers disagree with each other. The strength of the BN lies in the fact that it utilizes the correlation present between the classifiers, and is able to improve the classification performance even if the error rate of individual classifier falls below 0.5 This aspect of the approach becomes clear from the following theorem.

Consider two classifiers h_1, h_2 whose output is fed to the BN ($y, h_1, h_2 \in \{-1, 1\}$) shown in Figure 1 (a) and the output of the network is considered to be the output of the classification algorithm. Let the network be trained for the sample set $S = \{(x_1, s_1), ..., (x_n, s_n)\}$, where x is the input vector and s is corresponding true label ($\{-1, 1\}$). Then the probabilities of this BN, learned according to Maximum likelihood principle, are going to be

$$P(y = i|h_1(x), h_2(x)) = \frac{P(h_1(x), h_2(x)|s = i)P(s = i)}{P(h_1(x), h_2(x))} \tag{1}$$

where $i \in \{-1, 1\}$ and s is the true label for the input x. Now the following theorem holds-

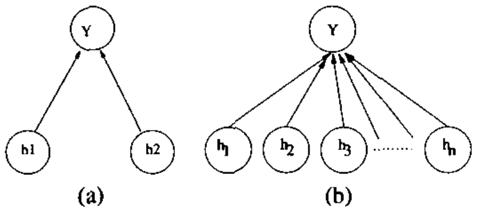

Figure 1. (a) Bayesian Network for combining two hypotheses, (b) Multiple hypotheses case.

Theorem 1 *The classification error (under absolute loss cost function) for the output of the network in Figure 1 (a), where the leaf node represents the output of the individual classifier, is less than or equal to the minimum of the classification error of h_1 and h_2. That is*

$$P(Y(x_i) \neq s_i|h_1(x_i), h_2(x_i)) \leq \min(P(h_1(x_i) \neq s_i), P(h_2(x_i) \neq s_i)) \tag{2}$$

Conditioned upon: 1) The output of the classifier s h_1, h_2 are statistically independent. 2) Perfect error-free classifier exists.

Before proving the theorem, we would like to give some basic definitions and will make the scenario more clear. Let us consider the case of binary classification where each classifier maps the data $h_i : x \in D \to \{0, 1\}$. Also consider that the data space D can be expressed as $D = \{D_0, D_1\}$ where, the perfect classifier (which is always correct) would map $x \in D_0 \to \{0\}$ and $x \in D_1 \to \{1\}$. By $h_1 = i$ we mean that given the sample, h_1 says that it has label "i". h_1, h_2 can also be seen as dividing the data space into four parts i.e. $D = \{D_{00}, D_{01}, D_{11}, D_{10}\}$ such that, if $x \in D_{ij}$ implies that $h_1 = i, h_2 = j$. Since h_1, h_2 are not perfect, they are going to make some errors during the classification. Let us express these as

$$\begin{array}{ll} P(h_1 = 1|D_0) = e_1; & P(h_1 = 0|D_1) = e_2; \\ P(h_2 = 1|D_0) = f_1; & P(h_2 = 0|D_1) = f_2; \end{array} \tag{3}$$

Namely, the probability of h_1 deciding "1" is e_1 when the data actually had the label "0". Also, independence of h_1, h_2 implies that $P(h_1 = i, h_2 = j|D) = P(h_1 = i|D) \cdot P(h_2 = j|D)$. Under these conditions, the decision $I(y|h_1, h_2)$ learned by the network in Figure 1 can be described as:

If $P(y = 1|h_1 = i, h_2 = j) > P(y = 0|h_1 = i, h_2 = j)$ then $I(y = 1|h_1 = i, h_2 = j) = 1$ otherwise $I(y = 1|h_1 = i, h_2 = j) = 0$.

Proof:

Let P_e denote the probability of error for the network shown in Figure 1, and Pe_1, Pe_2 denote the probability of error of hypothesis h_1, h_2 respectively. Then for N training samples:

$$\begin{aligned} Pe &= P(y = 1|D_0) \cdot P(D_0) + P(y = 0|D_1) \cdot P(D_1) \\ &= (\sum_i \sum_j (I(y = 1|h_1 = i, h_2 = j) \cdot \\ &\quad P(h_1 = i, h_2 = j|D_0)) \cdot P(D_0) + \\ &\quad (\sum_i \sum_j (I(y = 0|h_1 = i, h_2 = j) \cdot \\ &\quad P(h_1 = i, h_2 = j|D_1))) \cdot P(D_1) \\ Pe_1 &= P(h_1 = 1|D_0) \cdot P(D_0) + P(h_1 = 0|D_1) \cdot P(D_1) \\ &= P(D_0)e_1 + P(D_1)e_2 \\ Pe_2 &= P(h_2 = 1|D_0) \cdot P(D_0) + P(h_2 = 0|D_1) \cdot P(D_1) \\ &= P(D_0)f_1 + P(D_1)f_2 \end{aligned} \tag{4}$$

Depending upon the performance of the classifiers, we can have a number of cases relating the probability of data being "1" or "0" given the output of the individual hypothesis. Lets consider a particular case-

Case1:

$$\begin{array}{lcl} P(y = 1|h_1 = 1, h_2 = 1) & \geq & P(y = 0|h_1 = 1, h_2 = 1) \\ P(y = 1|h_1 = 0, h_2 = 1) & \leq & P(y = 0|h_1 = 0, h_2 = 1) \\ P(y = 1|h_1 = 1, h_2 = 0) & \leq & P(y = 0|h_1 = 1, h_2 = 0) \\ P(y = 1|h_1 = 0, h_2 = 0) & \leq & P(y = 0|h_1 = 0, h_2 = 0) \end{array} \tag{5}$$

Now from Equation 3,Equation 4 we get

$$\begin{aligned} Pe &= P(D_0)(e_1 f_1 + e_2 f_2 - e_2 - f_2) + e_2 + f_2 - e_2 f_2 \\ Pe_1 &= P(D_0)(e_1 - e_2) + e_2 \\ Pe_2 &= P(D_0)(f_1 - f_2) + f_2 \end{aligned} \tag{6}$$

Let $\Delta_1 = Pe_1 - Pe$ and $\Delta_2 = Pe_2 - Pe$. It can be easily shown (using simple algebra) that Δ_1, Δ_2 are non-negative. This proves that the error using the network shown in Figure 1 is less than or equal to the minimum of the two hypothesis.

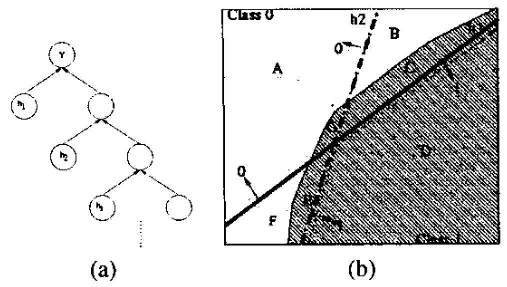

Figure 2. (a) Multiple hypotheses case, reconfiguring the network shown in Figure 1, (b) Graphical interpretation of the theorem.

This formalism can be extended to cases other than binary classification and to any number of base hypotheses. Figure 1 (b) shows network which would be used for combining "n" hypothesis by the Bayesian network framework. Instead of doing analysis directly on this network, we consider a simplified form shown in Figure 2(a) which has been obtained by introducing some extra nodes. In Figure 2(a), blank nodes represent the intermediate values (they are not observed and are introduced for the ease of analysis). At each blank node, the performance of the node (if the output of that node is interpreted as the classifier output) is going to be better than each of its children. This argument will hold recursively. In the end, the performance of the top node is going to be better than or equal to the best classifier in the set. A similar formalism can be given for the multiple class case. Consider that the output lies in 2^k dimensional space. Then the classifier can be thought of as a combination of k different classifiers. The analysis of the multiple hypotheses case can now be extended to take this into account.

We give the insight into the theorem by the aid of the graph shown in Figure 2(b). Consider the case of binary classification when the input feature vector x lies in a two dimensional space. The two classes are marked as "Class 0" and "Class 1", and the different shading shows the regions of each class in the feature vector space. An error free hypothesis (ground truth) is the non-linear boundary between the two regions. Let h_1 and h_2 be two linear hypothesis shown as a filled line and dotted line, respectively. Assume that the probability of error of each hypothesis is the class probability measure of the regions where the hypothesis makes the wrong decision. In this case, h_1 makes errors in regions F, G and C; h_2 makes errors in regions E, G and B. For simplicity, assume that the class probability measure of each of these regions is uniform. Hence, the area of each of the error regions is directly proportional to the classification error on those regions. For example, Figure 2(b) corresponds to the case stated in the proof of Theorem 1 in the following way: $C < B, E < F, G < A$. The new classifier, Y, as suggested by the Bayesian network, is depicted in Figure 2(b) as a thin dotted line. The error of Y now becomes the sum of the regions E, G and C. This is clearly smaller then the error made by the two hypothesis, as expected from Theorem 1. The theorem can be analyzed similarly for other cases. This mixture will not give any improvement if the area of the region B is same as C and of region E is same as F. In this case, if we relearn the hypothesis by weighing the incorrectly classified samples (i.e. the samples from one of the error regions), the new hypothesis will be different and an improved performance will be observed.

3 Generalization Performance

Now we show that the Bayesian network shown in Figure 1 (a) can also be represented as a decision tree shown in Figure 3 (a). This representation of BN (with binary random variables as its nodes) forms a complete binary tree of depth $d = n$ where n is the number of classifiers combined and the number of leaves is 2^n. This representation allows one to use the generalization bound presented in [5] for decision trees to bound the performance of new classifier in terms of the VC-dimension of the base classifiers. The theorem is repeated here for reference.

Theorem 2 *[5] For a fixed $\delta > 0$, there is a constant c that satisfies the following. Let D be a distribution on $X \times \{-1, 1\}$. Consider the class of decision trees of depth up to d, with decision functions in U. With probability at least $1 - \delta$ over the training set S (of size m), every decision tree T that is consistent with S has*

$$P_D[T(x) \neq y] \leq P_S[T(x) \neq y] + c \left(\frac{N_{eff} VCdim(U) \log^2 m \log d}{m} \right)^{1/3} \tag{7}$$

where N_{eff} is the effective number of leaves of T and VCdim(U) is the VC dimension of the base classifier U.

For the Bayesian network of classifiers with n base hypothesis, this reduces to

$$P_D[T(x) \neq y] \leq P_S[T(x) \neq y] + c \left(\frac{2^n VCdim(U) \log^2 m \log n}{m} \right)^{1/3} \tag{8}$$

as ($N_{eff} \leq 2^n$.) However, the number of effective leafs is going to be much smaller than 2^n, as such the bound is much smaller than what it appears at first glance. In a particular case when the base hypothesis is the linear perceptron, the VC dimension of the linear classifier is going to depend on the dimension of the feature vector.

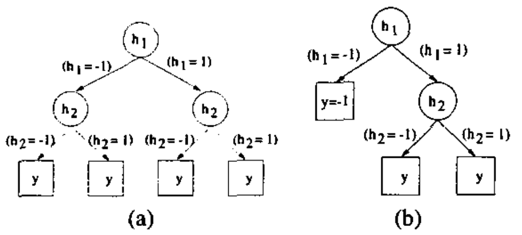

Figure 3. (a) Decision Tree representation of BN shown in Figure 1 (a), (b) Pruned decision tree

An optimal classifier will try to maximize the classification performance while minimizing the complexity. We show that one way to achieve this is by directly constructing the decision tree in an intelligent way (instead of the BN.) The algorithm for constructing the decision tree can be outlined as - Let $h_1, h_2, ..., h_m$ be the set of m classifiers. We want to make a decision tree which has these classifiers as its nodes and the value of the leaf node represents the output. Let $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ be the set of training samples. The first step is to evaluate each classifier on the training sample. For each classifier two different errors are measured- $e^m_{+1} = P(y_i = -1|h_m(\mathbf{x}_i) = 1)$ and $e^m_{-1} = P(y_i = 1|h_m(\mathbf{x}_i) = -1)$. The classifiers are ordered based on their performance on the training data (i.e. errors e^m_{+1}, e^m_{-1}). The classifier with the lowest error is picked to form the root node of the decision tree. If e^k_{-1} is the lowest, then h_k forms the top node of the decision tree. The output of the decision tree is '-1' whenever h_k says '-1' else the output is based on the decision made by the right sub-child of the root node. This suggests that the left child of the top node is a leaf with value '-1'. The right node points to a decision tree made of the classifiers. To obtain the classifier that forms the root node of the right subtree, the algorithm is repeated. If this time e^k_{+1} is the smallest then the right sub-child of the decision tree is also a leaf with value '1'. However this time the sample set is $S' = \{(x_i, y_i)\}$ for all 'i' such that $h_k(\mathbf{x}_i) = 1$. This process is repeated until all the classifiers are incorporated in the decision tree. We refer to the decision tree obtained in this way as the pruned decision tree. An example of such a pruned decision tree with two base classifiers (h_1, h_2) is shown in Figure 3 (b).

Corollary 1 *The classification error (under absolute loss cost function) for the output of the decision tree shown in Figure 3 (b), where the leaf node represents the output of the individual classifier, is less than or equal to the minimum of the classification error of h_1 and h_2. This holds provided the output of the classifiers is independent of each other given the data and a perfect error free classifier exists.*

This comes as an extension to the theorem 1 and is obvious from the algorithm (the way this pruned decision tree is constructed). Now the number of effective leafs is less than or equal to $n + 1$ where n is the no. of classifiers combined. The depth of the tree remains to be n. Hence the Equation 8 reduces to

$$P_D[T(x) \neq y] \leq P_S[T(x) \neq y] + c\left(\frac{(n+1)VCdim(U)\log^2 m \log n}{m}\right)^{1/3} \quad (9)$$

This improvement is achieved at the cost of reducing the performance on the training data (i.e. first term on the right side of the Equation 9.) It should be noted that this may not be the optimal tree, however, the results obtained using this method are quite promising.

4 Experiments and Results

We investigated the performance of the Bayesian Network of classifiers and the pruned decision tree by a series of experiments. First set of experiments was done on ten standard datasets from UCI ML repository and DELVE database. We targeted the binary classification problem. For this round of experiments, we choose linear perceptron as the base classifier and compared the performance of various classification techniques. In any technique for ensemble of classifiers, a big question is how to generate multiple classifiers. A number of techniques have been presented in [3]. The two techniques adopted by us are Adaboost and the sampling of the training examples. The learning algorithms were trained on part of the data and tested on the remaining data. Six fold crossvalidation was done for all the test cases. i. e., we divided the training examples into six groups. Five were used for training and the remaining one for testing. This process was repeated six times by varying the training and test sequences. All the results presented are on the test data. To learn the multiple classifiers using Adaboost, the classifier was learned on the training data. A weight vector was defined over the training samples. This learned classifier was used to classify the training samples (on which it was trained). The weight of the samples which were in error was increased where as that of others were decreased. The second classifier was trained on the same training samples with this new weight vector. This process was continued a number of times (here six.) To compare the performance with the adaboost, these classifiers were combined as suggested by Adaboost ([9]), using Bayesian network of classifier and pruned decision tree.

Table 1. Results on standard datasets. Each entry is -% correct classification (% error standard deviation), with Six classifiers, generated and combined using Adaboost, BNC-Ada:- generated using Adaboost and combined using Bayesian network of classifiers, Prune-Ada:- generated using Adaboost and combined using Pruned decision tree approach, BNC-samp:- generated by sampling the training examples and combined using Bayesian network of classifiers, Prune-samp:- generated by sampling the training examples and combined using Pruned decision tree approach

Dataset	perceptron	Adaboost	BNC-ada	Prune-ada	BNC-samp	Prune-samp
Breast Cancer	80.23 (4.02)	86.39 (3.55)	86.78 (3.82)	89.07 (3.39)	87.40 (2.11)	88.21 (2.28)
Credit	52.33 (4.72)	55.95 (3.18)	60.64 (3.67)	64.08 (3.65)	57.69 (2.32)	58.40 (2.30)
Heart	60.22 (7.68)	69.80 (5.48)	70.35 (6.23)	71.13 (6.08)	66.14 (4.31)	61.19 (3.56)
Ionosphere	84.46 (3.52)	83.88 (2.65)	85.46 (3.02)	86.03 (3.41)	85.92 (3.95)	84.97 (3.28)
Liver	53.08 (6.36)	64.57 (5.09)	66.57 (4.35)	66.28 (4.97)	63.74 (2.74)	59.88 (3.19)
Monk	56.47 (4.65)	59.83 (2.60)	64.03 (2.82)	64.87 (2.84)	63.37 (2.73)	63.03 (2.76)
Pima	60.91 (4.60)	62.37 (3.53)	63.62 (2.90)	62.84 (2.78)	65.58 (3.12)	64.86 (2.72)
Tic-tac-toe	56.13 (5.90)	62.09 (4.40)	66.49 (3.30)	68.17 (2.23)	65.41 (3.12)	62.36 (2.72)
Ring norm	65.66 (5.72)	72.25 (2.73)	72.97 (2.78)	72.50 (2.56)	72.54 (2.85)	72.46 (2.60)
Two norm	95.79 (0.52)	96.69 (0.42)	96.76 (0.40)	96.80 (0.36)	96.83 (0.12)	96.77 (0.59)

When generating multiple classifiers by sampling the training examples, the training samples were divided into six overlapping sets. A classifier was trained on each training set and were finally combined using both Bayesian network and the pruned decision tree. These combined classifiers were then tested on the remaining data (the one that was left out in the beginning).

Table 1 gives the classification results on the test data (left out data). It also gives the error standard deviation for each case and compares the performance between different algorithms. The improved performance, over both the single classifier and the Adaboost, demonstrates the power of these approaches. We see that at times the pruned decision tree performs even better than the Bayesian network of classifier. This can be attributed to its lower generalization error and hence more immunity towards overfitting. The second round of experiments was conducted on the real world problem of speaker detection.

4.1 Improving Multimodal Speaker Detection

Speaker detection [7, 4] is a fundamental problem experienced in any human-centered computer system. Detecting when a user is speaking to a computer (or any computerized appliance) has been classically achieved using audio sensors—microphones or microphone arrays. Recently a number of other sensors has become widely available, such as video cameras and ultrasound proximity detectors. With the growing number of sensors the challenge becomes in devising optimal techniques for fusion of the sensors' outputs. For instance, microphones and cameras can be used together to infer if a user is speaking to a computer. Visual cues are useful in deciding whether the person is facing the system and whether he is moving his lips. However, to distinguish an active user from an active listener, audio cues are also needed. Audio cues are not sufficient by themselves to discriminate a user speaking to the system from the same user speaking to another individual. This makes the problem of speaker detection multimodal in nature and the information from both audio, visual cues must be processed jointly to infer the users state.

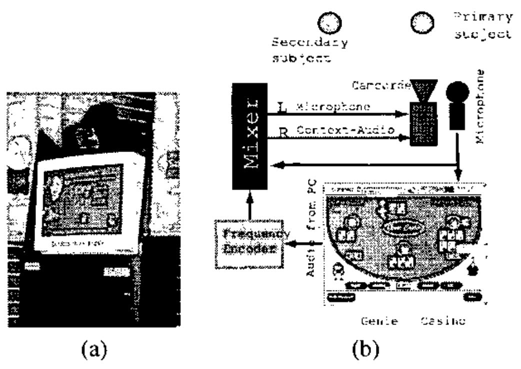

Figure 4. The Smart Kiosk (a) and Experimental setup for data collection (b).

The Smart Kiosk [2] 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. To detect the speaker we (see Figure 4) used a set of five "off-the-shelf" visual and audio sensors that process data acquired by a video camera and a microphone: a face detector [8], a Gaussian skin color detector, a face texture detector, a mouth motion detector, and an audio silence detector. Face detector uses a neural network module to detect the frontal face in the image. Skin color detector and texture detector are used to identify any skin color object present as a face or non-face. Mouth-motion detector, measures the movement of the lips and the audio detector, measures the energy in the audio signal present in the environment. All sensors output binary decisions.

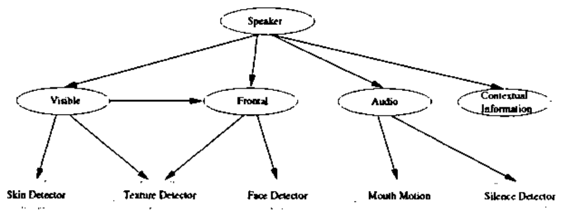

Figure 5. Bayesian Network for speaker detection.

The output of the five sensors is noisy and often ambiguous. Hence, one is required to the seamlessly integrate the sensors in order to achieve good detection performance. We have adopted a Bayesian network framework to solve this problem. Figure 5 depicts the Bayesian Network that reflects our expert knowledge of the domain and the sensors. The nodes "visible" and "frontal" represent the visual information extracted from the vision sensors. "Audio" node represents the information extracted from both vision (mouth motion detector) and the audio sensor (silence detector). These nodes are then integrated along with the application's contextual information to decipher the state of the user in the "speaker" node.

For testing the system, we collected five sequences of sensory data, of varying duration, totaling to 12500 frames. Figure 6 gives some video frames from a typical test sequence used. Each frame was hand labeled to provide the ground truth. The data was divided into training and testing sets. The parameters of the network in Figure 5 were learned from the training data. This network was then used to decode the labels for the training data. Multiple rounds of crossvalidation produced the correct classification in 84.76% of the frames.

Figure 6. Three frames from a test video sequence.

These, relatively poor results, motivated us to use the Bayesian mixture of classifiers in order to improve the detector's performance. Using the approach of Bayesian network of classifiers which combined four different base classifiers (here bayesian network) we achieved an improved performance of 89.6% correct classification. Multiple rounds of crossvalidation were done to check for overfitting. The standard deviation of the error was less than 2%. Table 2 shows complete results for this experiment. Results clearly indicate significant improvement in performance of the Bayesian mixture of classifiers over that of a single expert-designed network.

Table 2. Result on speaker detection dataset.

Dataset	Single Bayesian Network	Bayesian network of Classifiers
Speaker data	15.24 [% error]	10.4 [% error]

5 Conclusion and Discussion

In this paper we have argued that Bayesian network of classifiers provides a way to combine the output of the multiple experts. This method learns the correlation present between the classifiers and utilizes it to achieve better classification performance. We gave an algorithm to obtain pruned decision tree and results obtained on standard datasets demonstrated that a significant improvement in performance can be achieved while maintaining a lower bound generalization error.

The results on both the real world application and the standard datasets demonstrated the generalization performance of the classifier.

References

[1] L. Brieman. Stacked regression. Technical Report TR-367, University of California, Berkeley, 1994.

[2] 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.

[3] T. G. Dietterich. Machine-learning research. *AI Magazine*, 18:97–136, winter 1997.

[4] A. Garg, V. Pavlovic, J. Rehg, and T. S. Huang. Multimodal speaker detection using error feedback dynamic Bayesian networks. CVPR'00.

[5] M. Golea, P. Bartlett, W. S. Lee, and L. Mason. Generalization in decision trees and dnf: Does size matter? In *Neural Information Processing Seminar*, 1997.

[6] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:993–1000, 1990.

[7] J. M. Rehg, K. P. Murphy, and P. W. Fieguth. Vision-based speaker detection using bayesian networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 110–116, Ft. Collins, CO, 1999.

[8] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 203–208, San Francisco, CA, 1996.

[9] R. E. Schapire and Y. Singer. Improved boosting algorithms using cofidence rated predictions. To appear in Machine Learning.

[10] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.