Recognizing 100 Speakers using Homologous Naive Bayes

Hung-Ju Huang[1] and Chun-Nan Hsu[2]

[1] Department of Computer and Information Science,
National Chiao-Tung University,
Hsinchu City 300, Taiwan
hungju@cis.nctu.edu.tw

[2] Institute of Information Science, Academia Sinica,
Nankang 115, Taipei City, Taiwan
chunnan@iis.sinica.edu.tw

Abstract. This paper presents an extension of the naive Bayesian classifier, called *"homologous naive Bayes (HNB),"* which is applied to the problem of text-independent, close-set speaker recognition. Unlike the standard naive Bayes, HNB can take advantage of the prior information that a sequence of input feature vectors belongs to the same unknown class. We refer to such a sequence a *homologous set*, which is naturally available in speaker recognition. We empirically compare HNB with the Gaussian mixture model (GMM), the most widely used approach to speaker recognition. Results show that, in spite of its simplisity, HNB can achieve comparable classification accuracies for up to a hundred speakers while taking much less resources in terms of time and code size for both training and classification.

1 Introduction

The naive Bayesian classifier is a simple yet robust technique that generally performs well in a wide variety of application domains. This paper presents a variant of the naive Bayesian classifier, called *"homologous naive Bayes (HNB),"* which is applied to the problem of text-independent, close-set speaker recognition. The key difference between HNB and the standard naive Bayes is that HNB can take advantage of the prior information that a sequence of input feature vectors belongs to the same unknown class. We refer to such a sequence a *homologous set*, which is naturally available in speaker recognition. HNB classifies one homologous set at a time while the standard naive Bayes classifies one feature vector at a time. Derived from the Bayes rule and independence assumptions, HNB is designed to deliberately take advantage of homologous sets in test data. We compare HNB with some other straightforward extensions of the standard naive Bayes such as majority vote to classify homologous sets. Results show that HNB outperforms these extension methods and requires much less query vectors in the input homologous sets to significantly improve the classification accuracies.

We also empirically compare HNB with the Gaussian mixture model (GMM), the most widely used approach to speaker recognition. Results show that HNB can achieve comparable classification accuracies for up to a hundred speakers while taking much less resources in terms of time and code size for both training and classification. A remarkable point is that in our experiments, we discretize the input continuous features into ten bins in both training and classification. In our previous work [1], we give an explanation on why discretization works better than parameter estimation for the naive Bayes with continuous inputs. The results here provide additional evidence to our point.

2 Naive Bayesian Classifiers

A naive Bayesian classifier classifies a query vector $\mathbf{x}$ by selecting class c that maximizes the posterior probability

$$p(c|\mathbf{x}) \propto p(c) \prod_{x \in \mathbf{x}} p(x|c), \tag{1}$$

where x is a variable in $\mathbf{x}$, $p(x|c)$ is the *class-conditional density* of x, given class c. If x is a continuous variable, a conventional approach is to assume that $p(x|c) = N(x; \mu_c, \sigma_c^2)$, where $N(x; \mu_c, \sigma_c^2)$ is the probability distribution function of a normal distribution. In this case, training involves learning the parameters μ_c and σ_c from the training data [2]. This approach has been shown to be less effective than discretization when $p(x|c)$ is not normal, and discretization is often used. Generally, discretization involves partitioning the domain of x into k intervals as a pre-processing step. Subsequently, we can treat x as a discrete variable with k possible values.

In our previous work [1], we had explained why well-known discretization methods, such as entropy-based [3], bin-log$l$ and ten-bin, work well for naive Bayesian classifiers with continuous variables, regardless of their complexities. In addition, we identified conditions when discretization may hurt the classification performance. In summary, to avoid performance degradation, a discretization method should partition the domain of a continuous variable into intervals such that their cut points are close to decision boundaries to minimize the distortion due to the discretization, and their width should be large enough to cover sufficient examples. Since a naive Bayes takes all variables into account simultaneously, the impact of a "wrong" discretization for one variable can be absorbed by other variables under zero-one loss performance measure. Also, due to the way that a naive Bayes learns, correct classification of a given example depends only on the interval containing the values of that example, and is completely independent of how other regions are discretized. As a result, a wide variety of discretization methods can have similar performance in general.

3 Classifying a Homologous Set

We start by showing that a classifier must deliberately take advantage of the knowledge that all data have the same unknown class label; otherwise the knowledge will not improve the expected accuracy, and this is the case in general.

Consider a classifier which classifies one query vector into one of the n classes with accuracy σ. Suppose m query vectors were classified individually with this classifier. The expected value of the accuracy to classify these m query vectors is still σ. This can be derived by the expectation formula [4] as follows.

$$E_1 = \sum_{i=0}^{m-1} \frac{m-i}{m} \frac{m!}{(m-i)!i!} \sigma^{m-i}(1-\sigma)^i = \sigma. \tag{2}$$

On the other hand, suppose we have the information that the m vectors actually belong to the same class and classify them together (taking them as one object) using the same classifier. The expected value of the accuracy is:

$$E_2 = \frac{m}{m}\binom{1}{1}\sigma(1-\sigma)^0 + \frac{0}{m}\binom{1}{0}\sigma^0(1-\sigma) = \sigma. \tag{3}$$

The expected values of the above two cases are the same. This implies that the prior information that the query vectors come from the same class does not automatically improve the accuracy.

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a homologous set of query vectors with the same unknown class label c and each query vector $\mathbf{x}_t$ in $\mathbf{X}$ has r features $(x_{t1}, \ldots, x_{tr})$. We assume that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are drawn independently, and the symbol O denotes the prior information that all members in the homologous set have the same unknown class label. According to the Bayesian decision theory, we should classify this homologous set by selecting the class c that maximize $p(c|\mathbf{X}, O)$, the probability of c given $\mathbf{X}$ and O. We can derive three straightforward "extension" methods to estimate $p(c|\mathbf{X}, O)$ for classifying the query vectors in a homologous set. One method is by voting, which uses naive Bayes to classify each member in the homologous set and selects the class label predicted most often. Another is "Avg" method, which estimates $p(c|\mathbf{x}_t)$ for each $\mathbf{x}_t \in \mathbf{X}$ and averages the results to obtain $p(c|\mathbf{X}, O)$. The "Max" method estimates $p(c|\mathbf{x}_t)$ for each $\mathbf{x}_t \in \mathbf{X}$ and selects the maximum probability as $p(c|\mathbf{X}, O)$. The classification rule of the above methods is to pick the class c that maximizes $p(c|\mathbf{X}, O)$. The above methods are based on the idea that we can combine the estimation of $p(c|\mathbf{x}_t)$ to obtain $p(c|\mathbf{X}, O)$, by averaging or by selecting the maximum values. In fact, we can derive a method purely from the Bayes rule and independence assumptions as follows:

$$\begin{aligned}
p(c|\mathbf{X}, O) &= p(c|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, O) && (4)\\
&= \frac{p(c)p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, O|c)}{p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, O)} && (5)\\
&\propto p(c)p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, O|c) && (6)\\
&= p(c)p(\mathbf{x}_1|c)p(\mathbf{x}_2|c, \mathbf{x}_1)p(\mathbf{x}_3|c, \mathbf{x}_1, \mathbf{x}_2)\ldots \\
&\quad p(\mathbf{x}_n|c, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-2}, \mathbf{x}_{n-1})p(O|c, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n) && (7)\\
&= p(c)p(\mathbf{x}_1|c)p(\mathbf{x}_2|c)\ldots \\
&\quad p(\mathbf{x}_{n-1}|c)p(\mathbf{x}_n|c)p(O|c, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n) && (8)
\end{aligned}$$

$$= p(c)p(\mathbf{x}_1|c)p(\mathbf{x}_2|c)\dots p(\mathbf{x}_{n-1}|c)p(\mathbf{x}_n|c) \quad (9)$$
$$= p(c)p_c(\mathbf{x}_1)p_c(\mathbf{x}_2)\dots p_c(\mathbf{x}_{n-1})p_c(\mathbf{x}_n). \quad (10)$$

We can simplify (7) to (8) because of the assumption that all members in $\mathbf{X}$ are drawn independently[1]. Since O is logically implied by the event that $\mathbf{x}_1, \dots, \mathbf{x}_n$ are of class c, $p(O|c, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n) = 1$ and we can reduce (8) to (9). For the sake of being concise, we transit (9) to (10). Finally, since the naive Bayes assumes that all features are independent given class c, Equation (10) can be decomposed by the following equations:

$$(10) = p(c)p_c(x_{11}, x_{12}, \dots, x_{1r})p_c(x_{21}, x_{22}, \dots, x_{2r})\dots$$
$$p_c(x_{n1}, x_{n2}, \dots, x_{nr}) \quad (11)$$
$$= p(c)p_c(x_{11})p_c(x_{12})\dots p_c(x_{1r})p_c(x_{21})\dots p_c(x_{n(r-1)})p_c(x_{nr}) \quad (12)$$
$$= p(c)\prod_{t=1}^{n}\prod_{l=1}^{r} p_c(x_{tl}). \quad (13)$$

Therefore, the classification rule is to pick class c that maximizes (13) . We call this method "HNB" (Homologous Naive Bayes). Note that the training procedure of the above methods remains unchanged as the standard naive Bayes.

To evaluate the methods described above, we select fifteen real data sets from UCI ML repository [5] for our experiments. In the first experiment, we investigate the performance of the methods when there are three elements in a homologous set. We reported the average of the accuracies after running five-fold cross-validation on each data set. Table 1 gives the result. The result in Table 1 reveals that HBN significantly outperforms all other methods in all the data sets.

A large homologous set increases the chance that a naive Bayes correctly classifies a majority of the query vectors, thus ameliorating the performance of voting. In order to compare the performance of HNB and voting with different sizes of homologous sets, we selected a large data set ("Waveform", size = 5000) and repeated the same procedure as in the first experiment for different sizes of homologous sets (from one to 100). Note that when the size of homologous sets is one, both HNB and voting are reduced to a standard naive Bayes. This case serves as the base line for performance evaluation. Figure 1 plot the resulting curves, which show that the performance of voting improves as the size of homologous sets increases, but the curves of HNB grow faster. We repeat the same experiment for several other standard data sets from UCI repository and obtain similar results.

4 Application in Speaker Recognition

Speaker recognition is particularly relevant to our case because a large number of query vectors can be extracted from a short speech sentence, and obviously

[1] A feature vector may depend on other feature vectors in real situations if they come from the same class.

Table 1. Average accuracies of the naive Bayes using different extension methods when the size of a homologous set is three.

Data set	HNB	Voting	Avg	Max
australian	**97.16**	94.40	93.88	78.58
breast	**99.27**	98.42	98.36	93.27
chess	**94.19**	93.09	90.38	85.05
crx	**95.00**	92.61	92.75	91.16
german	**86.02**	81.88	76.22	71.79
glass	**80.71**	68.29	70.00	65.41
heart	**95.00**	93.03	89.09	85.61
hepatitis	**95.00**	86.33	94.00	91.00
iris	**100.0**	**100.0**	**100.0**	**100.0**
letter	**96.85**	86.42	91.24	88.03
pima	**88.39**	85.74	78.99	74.32
sonar	**93.95**	82.89	91.05	75.52
vehicle	**73.44**	66.75	71.25	60.12
waveform	**94.61**	77.71	87.45	79.42
wine	**100.0**	98.63	99.09	98.63
Average	92.64	87.07	88.25	85.53
Best	15	1	1	1

those vectors must come from the same speaker. A speaker recognition system should take advantage of this information.

In recent speaker recognition evaluations carried out by the National Institute of Standards and Technology (NIST) [6], the best GMM [7] based systems have outperformed the HMM [8] based systems. In this section, we will briefly review the Gaussian mixture model approach to speaker recognition and report our preliminary results on the empirical comparison between HNB and GMM in a text-independent, close-set speaker recognition task.

4.1 Gaussian Mixture Model

A Gaussian mixture density is a weighted sum of M component densities and is given by the form [7]:

$$p(w|\lambda) = \sum_{i=1}^{M} c_i b_i(w), \tag{14}$$

where w is a d-dimensional random vector, $b_i(w)$, $i = 1, ..., M$ is the component density and c_i, $i = 1, ..., M$ is the mixture weight. Each component density is a d-variate Gaussian function of the form:

$$b_i(w) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}} exp\{-\frac{1}{2}(w - \mu_i)' \Sigma_i^{-1} (w - \mu_i)\}. \tag{15}$$

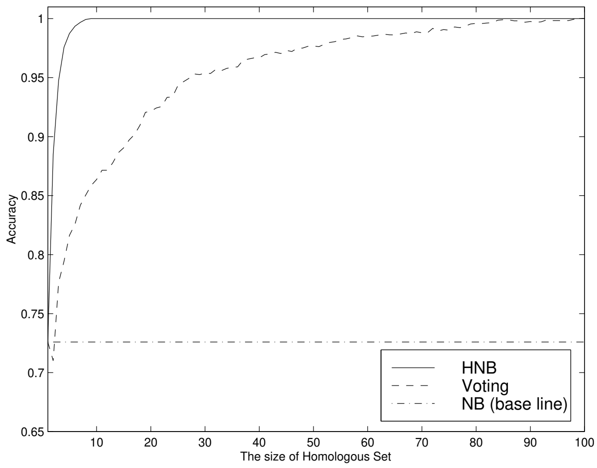

Fig. 1. Accuracies for classifying homologous sets with different sizes for the data set "Waveform".

with mean vector μ_i and covariance matrix Σ_i. The mixture weights must satisfy the constraint that:

$$\sum_{i=1}^{M} c_i = 1. \tag{16}$$

The complete Gaussian mixture density is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation:

$$\lambda = \{c_i, \mu_i, \Sigma_i\}, \qquad i = 1, ..., M. \tag{17}$$

Then each speaker is represented by a GMM and is referred to by his/her model parameter set λ. A GMM parameter set λ for a speaker is estimated using the standard Expectation Maximization (EM) algorithm [9]. GMM can classify homologous sets as follows. For a sequence of T query vectors $X = \{x_1, x_2, ..., x_T\}$ for the same speaker, the GMM log-likelihood can be written as:

$$f(X|\lambda) = \sum_{t=1}^{T} \log p(x_t|\lambda). \tag{18}$$

In the standard identification approach, the test speaker i^* is recognized from a set of S speakers by:

$$i^* = \arg\max_{i=1}^{S} f(X|\lambda_i). \tag{19}$$

Several system parameters must be tuned for training a Gaussian mixture speaker model, but a good theoretical guide for setting the initial values of those parameters has not been found. The most critical parameter is the order M of the mixture. Choosing too few mixture components can produce a speaker model that cannot accurately model the distinguishing characteristics of a speaker's distribution. Choosing too many components reduces performance when there are a large number of model parameters relative to the available training data [7].

4.2 Experiments

The database of the experiments reported in this paper is a subset of the TCC-300, Mandarin Chinese maintained by many research institutes in Taiwan. We used the speech data recorded at the National Chiao-Tung University. All speech signals were digitally-recorded in a laboratory using a personal computer with a 16-bit sound blaster card and a head-set microphone. The sampling rate was 16 kHz. A 30-ms Hamming window was applied to the speech every 10 ms, allowing us to obtain one hundred feature vectors from one second speech data. For each speech frame, both a 12th-order linear predictive analysis and a log energy analysis were performed. Firstly we filtered the feature vectors whose log energy value was lower than 30. This can be viewed as a silence removing process. A feature vector for training or testing contains the twelve linear predictive parameters. More than ten sentences are recorded from each subject speaker, and from each sentence, more than four thousand feature vectors are extracted. The evaluating procedure is described as follows: We randomly select five sentences for each speaker, one for testing and the others for training. Then we randomly select one thousand feature vectors from each sentence. Hence, for each speaker there are four thousand feature vectors for training and one thousand feature vectors for testing. There are several parameters must to be tuned for training a Gaussian mixture speaker model. $M = 32$ for GMM is used in our experiments, this setting is suggested in [7] and works well in our data sets. In contrast, no parameter tuning is necessary for HNB.

We run a five-fold cross-validation on different numbers of speakers and different homologous set sizes. One hundred query vectors represent one second speech data. We report the average accuracies, the average CPU time for training and classification in CPU timer ticks. Table 2 gives the results, which show that GMM uses smaller sizes of test vectors than HNB to reach a high accuracy. However, HNB takes much less resources in terms of time and code size for both training and classification.

More precisely, the average training speed of HNB surpasses GMM by about 329 times, and the average classification speed by about 15.73. It take about 0.5 seconds of additional test data on average for HNB than GMM to reach high accuracies ($> 99\%$). Since one more additional second of speech may not incur too much a handle to the speakers, this seems to be a tolerable or even favorable trade-off we can take in favor of HNB.

5 Conclusions and Future Work

HNB appears to be a promising approach to speaker recognition according to our preliminary study. We plan to purchase standard speaker recognition benchmark data sets from LDC and NIST to perform a thorough evaluation of our approach.

Acknowledgements

The speech data set was the courtesy of the Speech Processing Laboratory, Department of Communication Engineering, National Chiao Tung University, Taiwan.

The research reported here was supported in part by the National Science Council of ROC under Grant No. NSC 89-2213-E-001-031.

References

1. Hsu, C.N., Huang, H.J., Wong, T.T.: Why discretization works for naive bayesian classifiers. In: Machine Learning: Proceedings of the 17th International Conference (ML 2000), San Francisco, CA (2000)
2. John, G., Langley, P.: Estimating continuous distributions in Bayesian classifiers. In: In Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI '95). (1995) 338–345
3. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous valued attributes for classification learning. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '93), Chambery, France (1993) 1022–1027
4. Ross, S.: A First Course in Probability. Prentice Hall (1998)
5. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
6. Przybocki, M.A., Martin, A.F.: NIST speaker recognition evaluation. In: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France (1998)
7. Reynolds, D.A., Rose, R.C.: Robust text-independent speaker identification using gaussian mixture speaker models. IEEE Transactions on Speech and Audio Processing **3** (1995) 72–83
8. de Veth, J., Bourlard, H.: Comparision of hidden markov model techniques for automatic speaker verification in real-world conditions. Speech Communication **17** (1995) 81–90
9. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society **Series, B (39)** (1977) 1–38

Table 2. Results of the application in speaker recognition

Size of a homologous set	Average Classification Accuracy (in %) number of speakers					
	16		30		100	
	HNB	GMM	HNB	GMM	HNB	GMM
1	31.43	52.31	24.31	44.23	12.30	30.57
20	92.90	99.42	94.43	99.68	86.74	99.54
50	97.93	100.0	98.93	99.93	96.85	99.94
100	99.13	100.0	99.53	100.0	98.42	99.98
200	99.50	100.0	99.60	100.0	98.88	100.0
300	99.58	100.0	99.78	100.0	99.13	100.0
400	100.0	100.0	99.85	100.0	99.20	100.0
500 (5 sec)	100.0	100.0	100.0	100.0	99.40	100.0

number of speakers	Average Training Time (in CPU ticks) HNB	GMM	speedup
16	382.22	128056.44	335.03
30	716.22	240300.22	335.51
100	2446.85	776147.26	317.20
average	–	–	329.19

Size of a homologous set	Average Classification Time (in CPU ticks) number of speakers					
	16		30		100	
	HNB	GMM	HNB	GMM	HNB	GMM
1	0.19	3.88	0.38	5.867	6.55	101.59
20	3.57	50.92	6.84	103.96	115.94	1767.48
50	8.71	122.95	16.40	251.15	280.40	4109.76
100	16.90	242.50	31.70	500.31	542.16	8375.25
200	32.80	485.12	61.53	992.09	1049.73	16764.13
300	48.54	728.04	90.91	1490.00	1543.33	23206.66
400	64.50	967.39	120.66	1983.46	2043.27	30950.51
500 (5 sec)	80.00	1223.64	149.93	2484.63	2534.53	38834.37