Sparse Bayesian Approach to Classification

Zhixiong Chen, and Hao Tang, Member, IEEE

Abstract — This paper reports our recent efforts in the attempt to apply the Relevance Vector Machine (RVM) to text-independent speaker recognition tasks. The RVM represents a Bayesian extension of the widely applied Support Vector Machine (SVM), one of the leading approaches to pattern recognition and machine learning. Both the SVM and the RVM use a linear combination of kernel functions centered on a subset of the training data to make regressions or classifications. In the SVM, the number of vectors in the subset grows linearly with the size of the available training data, while in the RVM, only the most relevant vectors will be captured. So the RVM yields a much sparser approximation of the Bayesian kernel than the SVM. Our preliminary experimental results show that the RVM overall outperforms the SVM on speaker recognition while being advantageous over the latter for its exceptionally sparse nature, classification accuracy, and Bayesian probabilistic framework. Comparisons are also made for the Gaussian Mixture Model (GMM), a widely used non-discriminative approach to speaker recognition.

Index Terms — Sparse Bayesian Kernel, Pattern Classification, Relevance Vector Machine, Speaker Recognition, Machine Learning, Intelligent System

I. INTRODUCTION

U
nder supervised learning, we are given a set of training vectors \( X = \{x_i\}_{i=1}^N \) as input together with the corresponding target values \( T = \{t_i\}_{i=1}^N \). The objective is to make predictions of \( t \) for any given new vector \( x \).

The well-known Support Vector Machine (SVM) [6] seeks the point estimates of the weights \( W = \{w_j\}_{j=0}^N \) such that the linear combination of the base functions \( \Phi = \{K(x,x_j)\}_{j=1}^N \) fits the training targets as close as possible. At the same time, it reduces the computational complexity by forcing the majority of the weights to be zero. In other words, the SVM selects only a subset of \( X \), called the Support Vectors, in regression or classification. We can write the prediction target value as follows:

\[
t(x; w) = w_0 + \sum_{j=1}^N w_j K(x,x_j)
\]

Here, \( K(\cdot, \cdot) \) is a bivariate kernel function.

In the case of classification, the target values are class labels that belong to a discrete set. The linear combination of kernel functions will give the largest margin separation boundary between classes in a feature space generated by a Mercer kernel function.

The SVM framework avoids over-fitting by performing classification with a maximum margin hyperplane and reduces computational complexity by making use of only a small subset of the training data. It is a leading approach to do regression and classification, and has already been successfully applied to a wide variety of problems such as fraud detection, bio-informatics, text mining, etc [15, 6]. Even so, the SVM is not a fully probabilistic model. It generates hard point estimates instead of probabilistic distributions. The Mercer kernel restriction in order to generate a sparse model is also limiting. Moreover, both theoretical and practical reasons still justify the effort of searching for methods that produce even sparser models [7, 11].

Recently, a sparse Bayesian method, called the Relevance Vector Machine (RVM), was proposed [7]. It is a discriminative modeling technique in a form very similar to the SVM, but it is a fully probabilistic model claimed by the authors that shows greater sparseness and higher accuracy than the SVM. This is achieved through the framework of sparse Bayesian learning and the introduction of prior distributions over the predictions of the weights.

At the moment, unlike the SVM, the RVM has only been implemented with a few basic routines in MatLab. There has been no complete, practically usable implementation yet. Tipping has given one implementation in MatLab [12]. But it cannot handle high dimensional data with a large number of vectors due to the computing resource in our experiments. In [8], a new approach called “fast algorithm based on marginal likelihood maximization” was proposed. The difference is that this new approach constructs the model by gradually adding/adjusting the relevance vectors instead of pruning the whole training set to select the relevance vectors. Since we have not seen the implementation publicly, we would like to implement the algorithm in MatLab first and then in other languages to make it more practically usable.

Our first application of the RVM is to speaker recognition. As we know, in highly secure applications such as service
access control and telephone banking, biometric technology has been playing an increasingly important role [1]. In addition to fingerprints, eyes, face, and handwriting, human voice is another popular biometric characterized by its moderate cost and easy-to-use, unforgettable, and unobtrusive nature. Speaker recognition is a computer-based task that answers the questions of “who are you” and “are you whom you claim to be” based on a person’s voice, and is one of the few recognition areas where machines can outperform humans [2]. According to whether or not the text being spoken is known a priori, speaker recognition can be either text-dependent or text-independent.

In speaker recognition, speech features that can be extracted from a voice signal and that are able to show high speaker discrimination power, high inter-speaker variability and low intra-speaker variability are desired [3]. Speaker recognition is essentially a pattern classification process that employs proper speech features to produce discriminative information.

For the last couple of decades, text-independent speaker recognition technology has brought about many research and engineering efforts both in the academia and industry. A number of methods (classifiers) have been developed to tackle the speaker recognition problem. The Gaussian Mixture Model (GMM) is among the pioneer but most effective approaches for the speaker recognition problem. The Gaussian Mixture Model (GMM) is a generative statistical modeling technique that constructs the “best” decision boundary or hyperplane in the feature space through discriminative modeling technique that is capable of capturing the inter-speaker variability, and is still a prevailing method for speaker recognition today due to its great flexibility and precision in modeling speech features by multimodal probability densities. As a new pattern classification technique, the SVM has been applied to speaker identification and verification and has demonstrated promising results [5]. Different from the GMM, the SVM is a discriminative modeling technique that constructs the “best” decision boundary or hyperplane in the feature space through maximum margin criterion [6].

Our preliminary experimental results show that the RVM classifier overall outperforms the SVM and GMM on the speaker recognition task, although not that significance. It does employ far less feature vectors in the resulting model while employing proper speech features to produce discriminative information.

II. RVM CLASSIFICATION USING MARGINAL LIKELIHOOD MAXIMIZATION

The objective of a binary classification is to predict the posterior probability for a given input $x$ of class 0 or class 1. We can use the logistic sigmoid function $\sigma(y) = (1 + e^{-y})^{-1}$ to link the posterior probability as

$$P(t_n | y_n) = (\sigma(y_n))^t_n (1 - \sigma(y_n))^{1-t_n}.$$  

This is a two-class Bernoulli distribution of $t_n \in \{0,1\}$ for a given target value $y_n$, where $y_n = y(x_n, w)$ is defined by the weighted summation over the base kernels $\{ K(x_n, x_i) \}_{i=1}^{N}$ for a given set of training samples $\{x_n\}_{n=1}^{N}$, that is,

$$y(x_n, w) = \omega_0 + \sum_{i=1}^{N} \omega_i K(x_n, x_i).$$  

(1)

As we can see, the weights $w = \{ \omega_0, \omega_1, ... , \omega_N \}^T$ are going to be tuned in the training process and the majority of them will be zero. All the feature vectors with non-zero weights are called the Relevance Vectors. The kernel function $K(\cdot, \cdot)$ can be of the form of Gaussian, Euclidean distance, Laplacian, polynomial, etc. Unlike in the SVM, the kernel in the RVM does not have to be a Mercer kernel.

The likelihood of a set of observed labels $t = [t_1, t_2, ..., t_N]^T$ corresponding to $\{ x_n \}_{n=1}^{N}$ are then defined as

$$P(t | w) = \prod_{n=1}^{N} \sigma(y(x_n, w))^{t_n} (1 - \sigma(y(x_n, w)))^{1-t_n}.$$  

The argument now is to find the weight set $w$ such that the above $P(t | w)$ is maximized. Because of the discontinuity of the likelihood $P(t | w)$, the marginal likelihood cannot be obtained in a closed form. We must use an iteration process [8, 9]. The idea is to use Laplace’s approximation method based on the marginal likelihood maximization $L(\alpha)$ (see expression (6)). At each iteration, the RVM regression technique is used to find the local maximization and the weight set $\omega_{mp}$ over the given prior hyperparameters $\{ \alpha \}_{i=1}^{N}$. The hyperparameters are then updated using $\omega_{mp}$. The process keeps going on until some stop criteria are met. More specifically, we have $w_{mp}$ that can be expressed in the form

$$\omega_{mp} = \Sigma \Phi^T B \Sigma^{-1} \Phi^T B \Phi + A \Sigma^{-1}.$$  

Here,

$$\Sigma = \text{diag} \{ \alpha_0, \alpha_1, ..., \alpha_N \}^T$$  

is a hyperparameter diagonal matrix, and

$$B = \text{diag} \{ \beta_1, ..., \beta_N \}$$  

is another diagonal matrix with

$$\beta_n = \sigma(y(x_n))(1 - \sigma(y(x_n))).$$  

\Phi is an augmented kernel matrix defined as

$$\Phi = \{ \phi(x_1), ..., \phi(x_N) \}^T,$$  

with

$$\phi(x_i) = [1, K(x_i, x_1), ..., K(x_i, x_N)]^T.$$  

Using the new $w_{mp}$, the new target $\hat{t}$ is then obtained through

$$\hat{t} = \Phi \omega_{mp} + B^{-1}(1 - \sigma(y(x_n))).$$  

(5)
Following the defined weight vector $\omega_{mp}$ is the step to update the prior hyperparameters $\{\alpha_i\}_{i=1}^N$. As we know, the sparse Bayesian learning is formulated as the local maximization with respect to the hyperparameters $\{\alpha_i\}_{i=1}^N$ of the marginal likelihood:

$$L(\alpha) = -\frac{1}{2} \left[ N \log 2\pi + \log |C| + \mathbf{t}^T C^{-1} \mathbf{t} \right]$$

(6)

where

$$C = \sigma^2 I + \Phi A^{-1} \Phi^T$$

and $\sigma$ is the standard deviation of the noise in the samples. If we set

$$s_i = \phi(x_i)^T c^{-1}_{j=1} \phi(x_i), \quad q_i = \phi(x_i)^T c^{-1}_{j=1} t$$

and

$$C_{-i} = C - \alpha_i^{-1} \phi_i \phi_i^T,$$

then:

If $q_i^2 > s_i$, $L(\alpha)$ will have its local maximization. So the vector $x_i$ will be added to the model as a relevance vector and $\alpha_i$ is updated to a new value:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \quad \text{(7a)}$$

Otherwise, the maximization will be reached at infinity,

$$\alpha_i = \infty, \quad \text{(7b)}$$

The crucial observation is that the majority of values [8] (over 90% in our experiments) of $\{\alpha_i\}_{i=1}^N$ are infinity. Because the prior distribution for the weights is defined explicitly as

$$P(w | \alpha) = \frac{1}{\sqrt{(2\pi)^{N+1} ||A^{-1}||}} e^{-\frac{1}{2} w^T A w},$$

the distribution is peaked around zero when $\alpha_i$ is large enough. So the corresponding $\omega_i$ has to be set to zero. Therefore, we obtain quite few feature vectors in the resulting model, or in other words, obtain an extremely sparse Bayesian model.

### III. The Algorithm

In the RVM training, pruning method is used quite often [7]. But this technique suffers when the number of input vectors becomes larger and larger and the dimension of each vector increases, because the computation resource requirements increase dramatically. The worst thing is that this method eventually is no longer able to work because of its enormous computation resource consumption.

Instead of taking all the feature vectors as input at the very beginning and gradually pruning them from the model, we implement a new algorithm proposed in [8]. It starts with an empty model with zero relevance vectors, gradually adds feature vectors into the model, and sometimes deletes certain vectors from the model (deletion happens mostly toward the end of the training process). The idea is based on the marginal likelihood maximization [9,10].

The first feature vector (relevance vector candidate) is picked when the normal projection to vector $t$ reaches the maximum. That is, pick a vector $x_i$ from the training set such that $\|\phi_i^T t\|/\|t\|$ is maximized. Meantime, using formula (2), (3) and (7) to obtain the initial $w$ and $\{q_i, s_i\}_{i=1}^N$.

Once the initial values are set, we then repeat the following steps:

1. Modify the model and find the marginal change. If $q_i^2 > s_i$, add the vector $x_i$ to the model if it is not in there; otherwise ($q_i^2 \leq s_i$), either delete the vector if it is in the model or do nothing.
2. Refit the Laplace Approximation. With the updated model, we can obtain new estimated values of $y$, compute every $\beta_j$ through its definition, and get the new $i$ via formula (5).
3. Update the model. Based on the result of step 1, either update $\alpha_i$ using (7a) or set $\alpha_i$ to infinity.
4. Re-compute the weights $w$ and $\{q_i, s_i\}_{i=1}^N$.
5. Test convergence. The iteration shops when either the logarithm ratio of two consecutive $\alpha_i$ for all $\{\alpha_i\}_{i=1}^N$ or the change of marginal likelihood $L(\alpha)$ falls within a predefined range.

All the calculations are done in MatLab.

### IV. The Model and the Results

A simplified speaker recognition model is depicted in the following figure (Figure 1).

Figure 1. A simplified speaker recognition model

The first experiment is to apply the algorithm to the TIMIT (Texas Instruments Massachusetts Institute of Technology) speech database, which is widely available. Our real purpose is to apply the algorithm to real world scenarios with no perfect conditions on sound rooms, sessions and microphone as in the TIMIT database.

The TIMIT corpus consists of 630 male and female speakers from 10 different dialect regions in America, and there are 10 utterances for each speaker. The speech was recorded using a high quality microphone in a sound proof booth at a sampling frequency of 16 kHz, with no session interval between
recordings. It provides a perfect situation for speaker identification and verification for initial testing.

The speech features in our experiments were the 39-dimensional Mel-scaled Frequency Cepstral Coefficients (MFCC). MFCC is a very popular choice of speech features for speaker recognition. It consists of 13 absolute MFCC values known as the basic MFCC’s and the first and second order derivatives of these basic MFCC’s. Energy normalization and cepstral mean subtraction were used in the computation of the MFCC’s for the purpose of noise reduction and channel compensation.

Our experiments were focused on the classification accuracy, the number of “in-model” feature vectors (i.e., the relevance vectors or support vectors) and the relative training time of the models.

The first job is to convert the TIMIT corpus into MFCC files. An MFCC file is corresponding to an utterance 3 to 4 seconds long.

We carried out the training by choosing randomly one speaker, say A, and a combination of other four speakers, say X. For each chosen speaker, we picked randomly 2 MFCC files. The 10 MFCC files formed our training set. The class labels are either 1 or -1. For speaker A, the labels are 1, while for all other speakers, the labels are -1. We denote the model as model A. The rest of the MFCC files for all speakers were used as the testing data.

We say a model A gives a correct classification decision for a given test MFCC file if the number of positive values generated by the model is greater than that of negative values when the MFCC file is from speaker A, or if the number of negative values is greater than that of positive values when the MFCC file is from other speakers. It is not a per-frame voting scheme. We actually adopted a simple ad hoc majority voting scheme.

The same experiment was also carried out for the SVM [13] and GMM [14]. We repeated the experiment 25 times for different models. Table 1 gives the average accuracy rates. The accuracy for the GMM is a bit lower than expected. The reason might lie in the fact that the size of our chosen training set is too small.

![Table 1: Accuracy of GMM, SVM and RVM models](image)

Table 2 shows the huge difference in the sparseness of the resulting models between the SVM and RVM.

![Table 2: Percent of “in-model” feature vectors](image)

In the experiments, we found that although our algorithm increased the capacity of the number of vectors that can be handled [12], the speed was relatively slow. For example, it took about an average of 40 hours (due to 3 different PCs) to train a model with approximately 3000 vectors with 39 elements. In terms of time, it is not comparable to SVM at all.

V. CONCLUSION

From our preliminary experimental results presented in this paper, it is clear that the RVM technique has great potentials in classification tasks such as speaker recognition. The most evident and compelling results are its accuracy and sparseness. However, the biggest challenge ahead is to derive an effective algorithm to reduce the training time dramatically when the number of training vectors is very large. Probably the first step is to implement the algorithm using languages such as C or C++.

ACKNOWLEDGMENT

One of the authors, Zhixiong Chen, would like to thank the CAIP center, Rutgers University, NJ, for providing research opportunities during the summer visit from June to August, 2004, especially thank professor Jim Flanagan, Larry Rabiner, Sorin Dusan, and Aaron Rosenberg for their hospitality.

The authors also thank anonymous reviewers for their valuable comments.

REFERENCES