

Radial Basis Function Neural Networks for Speaker Verification

LI GUOJIE

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University in
fulfillment of the requirement for the degree of
Master of Engineering

2004

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other university or institution.

.....

Date

.....

Li Guojie

Acknowledgements

I would like to express my deepest gratitude and appreciation to my supervisors, A/Prof. P. Saratchandran and Prof. N. Sundararajan, for their enthusiastic encouragement, excellent guidance and the kind support during this research project and development of this thesis.

Special thanks are also dedicated to my colleagues and staff at Computer Control Lab at the School of Electrical and Electronic Engineering for their help and support.

I would also like to acknowledge the School of Electrical and Electronic Engineering in Nanyang Technological University for opportunity for this study, financial support, and research facilities.

Summary

This thesis presents the application of a minimal radial basis function (RBF) neural network, referred to as MRAN (Minimal Resource Allocation Network) for speaker verification. Extension of MRAN to elliptical basis functions has been studied too.

MRAN is a sequential learning algorithm for radial basis function neural networks. During the training, MRAN allows hidden neurons to be added or removed thus to realize a minimal network. MRAN recruits hidden neurons based on the novelty of the input data. If all of the novelty criteria can not be satisfied, the existing network parameters are updated by extended Kalman filter (EKF). Additionally, MRAN's pruning strategy removes hidden neurons from the network if their contributed output to the output layer is insignificant. In this way, MRAN is adapted to fit the dynamics of the input data closely.

In this research project, we used MRAN to perform speaker verification tasks and compared the results of MRAN with those of normal Radial Basis Function (RBF) networks, Elliptical Basis Function (EBF) networks. TIMIT corpus is used as the speech database in our experiment. The experiment involves several phases. In the first phase, utterances spoken by the speaker are translated into linear predictive coefficients (i.e., feature vectors). These feature vectors are then fed to the MRAN to train the network. After the network has been trained, its verification decision threshold is then determined. The verification threshold is used to make the decision

of accepting or rejecting identity claims based on the network output. In our experiment, we used this procedure to train 76 MRAN networks corresponding to 76 speakers from the TIMIT database. The experimental results showed that MRAN outperforms the conventional RBF network. Compared to elliptical basis function (EBF) neural networks, MRAN produces comparable error rates at a much lower complexity in term of number of network parameters. The computational complexity for each network is analyzed. The computational complexity for all the RBF and EBF networks is $O(H^2N + N^2)$, where H represents the total number of hidden neurons in the network and N represents the total number of training data. Compared with RBF and EBF networks, MRAN network requires much lower computational complexity, which is $O(H^2N)$.

We also extended MRAN to elliptical basis functions, referred to as MRAN-EBF. MRAN-EBF is used to perform speaker verification tasks and compared with MRAN. The experimental results show that the performance of MRAN-EBF with full covariance matrix is worse than that of MRAN. MRAN-EBF with diagonal covariance matrix produces comparable error rates with MRAN but with a higher complexity.

Contents

Acknowledgements	i
Summary	ii
List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Major Contributions	3
1.4 Organization of the Thesis	4
2 Review of Radial Basis Function Networks	6
2.1 Radial Basis Function Networks	7
2.2 Minimal Resource Allocation Networks	13
2.2.1 Introduction to MRAN	14
2.2.2 MRAN Applications	22
3 Review of Speaker Verification	24
3.1 Introduction to Speaker Verification	24
3.2 Acoustic Parameters.....	27
3.2.1 Pitch.....	28
3.2.2 Frequency	28

3.2.3 Linear Predictive Coefficients	29
3.3 Speaker Verification Methods	30
3.3.1 Long-Term Statistical Methods	30
3.3.2 Vector Quantization	32
3.3.3 Hidden Markov Models	35
3.3.4 Artificial Neural Networks	38
3.4 Database for Speaker Verification – TIMIT	40
4 MRAN for Speaker Verification	43
4.1 Overview	43
4.2 Speech Database and Feature Extraction	46
4.3 Speaker Enrollment	50
4.4 Speaker Verification.....	54
4.5 Decision Threshold	56
4.6 Experimental Results	58
4.7 Computational Complexity Analysis	61
4.7.1 RBF	62
4.7.2 EBF with Sample Covariance	64
4.7.3 EBFs with EM algorithm	65
4.7.4 MRAN.....	67
4.7.5 Comparison	68
5 Extension of MRAN to MRAN-EBF	70
5.1 MRAN-EBF.....	70
5.1.1 Gaussian Functions	72
5.1.2 From MRAN to MRAN-EBF.....	73
5.2 Application of MRAN-EBF to Speaker Verification	76
5.3 Experimental Results	80
6 Conclusion and Recommendations	82
6.1 Conclusion.....	82
6.2 Recommendations for Further Research.....	85

Author's Publications	87
Bibliography	88

List of Figures

Figure 2.1: Architecture of a typical RBF network	10
Figure 3.1: Speaker-based vector quantization system.....	33
Figure 3.2: A four-state HMM.....	36
Figure 3.3: Flow chart diagram for the training phase.....	37
Figure 4.1: Phases involved in the experiments.....	44
Figure 4.2: Network setup for each speaker in the speaker set.....	51
Figure 4.3: The training input data sequence.....	52
Figure 4.4: FAR and FRR versus the decision threshold of a MRAN network	57
Figure 5.1: Architecture of MRAN-EBF	71

List of Tables

Table 3.1: Dialect distribution of speakers in TIMIT	41
Table 3.2: TIMIT speech material	42
Table 4.1: Speaker sets used in the experiments.....	47
Table 4.2: Typical parameter settings	53
Table 4.3: Comparison of Results.....	59
Table 4.4: Notations used in complexity computation	61
Table 4.5: Comparison of computational complexity.....	68
Table 5.1: Speaker sets used in the experiments.....	76
Table 5.2: Typical parameters settings for MRAN-EBF and MRAN	77
Table 5.3: FAR's, FRR's and numbers of hidden neurons for MRAN, MRAN-EBF, and MRAN-EBF with diagonal covariance.	80
Table 6.1: Experimental results of MRAN	83
Table 6.2: Computational complexity of MRAN and other networks.....	84

Chapter 1

Introduction

1.1 Motivation

In recent years, neural computing has emerged as a practical technology with successful applications in many areas. Artificial neural networks (ANNs) are information paradigms inspired by biological nervous systems, such as human brain [39]. ANNs have remarkable ability to derive meaningful or mathematical models from complicated or imprecise data. ANNs have been successfully applied across a large range of problem domains, such as finance, medicine, engineer, physics and geology [21].

Over the years, various types of artificial neural networks have been developed. The earliest kind of neural network is a single-layer perceptron network. A perceptron network consists of a single layer of output nodes. The simple architecture of perceptron network limits its computational power. Multi-layered feed-forward network (MFN) is one of the most prominent neural networks. A MFN consists of

multiple layers of computational units, usually interconnected in a feed-forward way. This means that each neuron in one layer has directed connections to the neurons of the subsequent layer. For a classification problem, a MFN divides the pattern space using hyperplanes. An equally appealing and intuitive approach is to divide up space using circles or (more generally) hyperspheres, which is exactly what a radial basis function (RBF) network does. A RBF network only has one hidden layer of radial units. Compared with MFN, a RBF network has a simpler topology and it can be trained much faster [39].

The minimal resource allocation network (MRAN) is a sequential learning algorithm for RBF network [25]. MRAN allows hidden neurons to be added or removed as training progresses thus to realize a minimal network. In traditional RBF network training, the number of hidden neurons is often fixed a priori. Linear least square methods like LMS are used to estimate the weights connecting the hidden neurons to the output layer. One of the drawbacks of the traditional approaches is that it usually results in too many hidden neurons [1]. A sequential learning method for RBF network has been proposed by Platt [19]. In Platt's algorithm, hidden neurons are added dynamically based on the novelty of the network input data so that the number of hidden neurons does not need to be selected a priori. However, once a hidden neuron is allocated, it can not be removed. As a result, noise data patterns may trap the network to create redundant hidden neurons. MRAN overcomes this problem by adding a pruning strategy based on the relative contribution of each hidden unit to the overall network output. MRAN has been successfully applied in many areas such as function approximation, time-series prediction, etc [25][31].

In recent years, there has been a booming interest in the use of biometric characteristics as a means of recognizing and identifying a person. Human voice is one of the most important biometric identifiers of a person. While speech recognition is concerned with extraction of the linguistic message underlying a

spoken utterance, speaker recognition is concerned with extraction of the identity of the person speaking the utterance. The general area of speaker recognition can be categorized into speaker verification and speaker identification. The task of speaker verification is to determine from voice samples whether a person is whom he or she claims. On the other hand, speaker identification aims to determine which one of a set of known voices best matches the input voice samples. Both speaker verification and speaker identification can be constrained (text-dependent) or unconstrained (text-independent). A speaker verification system is said to be “text-dependent” if it knows that the speaker is supposed to say. On the other hand, a text-independent speaker verification system does not have foreknowledge of what the speaker is supposed to say.

In this project, we investigate various methods to perform text-independent speaker verification tasks.

1.2 Objectives

The main objective of this research project is to apply an existing sequential learning algorithm for RBF networks (named Minimal Resource Allocation Network, or MRAN) to speaker verification problems, and evaluate its performance, by comparing the results with those of other well-known methods. The project is also aimed to extend MRAN to elliptical basis functions (EBF) and compare normal MRAN with elliptical function based MRAN in the context of speaker verification.

1.3 Major Contributions

Major contributions made in this project include:

- Demonstration of the potential of MRAN for speaker verification. Text-independent speaker verification problems are investigated and the suitability of MRAN for speaker verification is shown. MRAN is used to perform text-independent speaker verification tasks. The comparison of MRAN with other methods such as normal RBF and elliptical basis function (EBF) networks is done, to show the advantage of using this sequential learning algorithm for speaker verification tasks. The experimental results show that MRAN outperforms RBF network and EBF networks in terms of geometric mean error and complexity (expressed in the number of network parameters). The average equal error rate of MRAN is much less than that of RBF, however, it is slightly larger than those of EBFs. In general, MRAN produces comparable error rates to other well-known methods but with much less computational complexity.
- Extension of MRAN to elliptical basis functions (EBF) (referred to as MRAN-EBF) and application of MRAN-EBF on speaker verification tasks. MRAN is a RBF based network, and we extend it to EBF based network – MRAN-EBF. Experiments have been done to compare the performance of MRAN-EBF with MRAN. The experimental results show that MRAN-EBF performs worse (in terms of false accept rate and false reject rate) than MRAN and with a higher degree of complexity. One of the reasons could be that the extended Kalman filter linearization produces highly unstable filters due to the high degree of nonlinearity of the problem.

1.4 Organization of the Thesis

This thesis is divided into six chapters. Chapter 2 introduces radial basis function networks in general, and minimal resource allocation network in particular. Chapter

3 gives a brief review of speaker verification. In chapter 4, MRAN is applied to speaker verification. The setup and the procedures of the experiments are outlined first. Speech data from the TIMIT corpus is used as the testing data. After the experiments have been completed, the experimental results of MRAN are then compared with those of other methods. The extension of MRAN to MRAN-EBF is introduced in Chapter 5. Conclusions and the scope for future work are presented in Chapter 6.

Chapter 2

Review of Radial Basis Function Networks

Artificial Neural Networks (ANNs) are information paradigms inspired by the way biological nervous systems, such as human brain, process information. An ANN consists of a large number of densely connected processing units, which are often referred to as artificial neurons or nodes. These nodes are interconnected through links. Each link is associated with a connection strength, which is often called synapse strength or weight. The training of the network involves determining various parameters such as the center vectors and the weights.

The first computational model for ANN – single-layer perceptron network was proposed by McCulloch and Pitts in 1943. A single-layer perceptron network consists of only a single layer of output nodes. The inputs are fed directly to the output via a series of weights. Such simple perceptron networks are only capable of learning linearly separable patterns. Significant progress on ANN has been achieved since 1980's. The architectures of ANN evolve from the perceptron to complex structures such as multi-layer feed-forward networks (MFN) and recurrent networks.

In our study, we focus on a special kind of ANN called Radial Basis Function (RBF) networks.

From the view of topology, a RBF network is a special case of the multi-layer feed-forward neural networks. A RBF and a general multilayer feed forward neural network differ on the node characteristics and the training algorithm. Sigmoidal functions are usually used as node characteristics for multi-layer feed-forward networks, while radial basis functions are employed as node characteristics for RBF networks. Additionally, Back-Propagation (BP) is the most popular training algorithm for multilayer feed forward networks, while different training algorithms are used for RBF networks. The following section presents the architecture of RBF networks and various training algorithms for RBF networks.

2.1 Radial Basis Function Networks

Radial basis function networks have their origins in methods for performing exact interpolation of a set of data points in a multi-dimensional space [27]. In exact interpolation problem, every input vector must be mapped exactly onto its corresponding target vector.

The data interpolation problem can be stated as follows: Given a set of d -dimensional input space $\{\mathbf{x}_i\}$ and a set of corresponding one-dimensional target space y_i , the exact interpolation problem is to find a function $f(\cdot)$ which satisfies:

$$y_i = f(\mathbf{x}_i) \quad \forall i \quad (2.1)$$

The approach that radial basis function takes is to form a set of basis functions, one for each data set. The radial basis function is of the form $\Phi(\|\mathbf{x} - \mu\|)$, where $\Phi(\cdot)$ is a

non-linear function, \mathbf{x} is the input pattern and μ is the center of the function. Obviously, the k^{th} function depends on the distance $\|\mathbf{x} - \mu\|$ (in the Euclidean sense). Another important property of radial basis function is that its output is symmetric around the associated center μ . Thus $f(\mathbf{x}_i)$ can be taken to be a linear combination of the outputs of all the basis functions:

$$f(x) = \sum_k \omega_k \Phi(\|\mathbf{x} - \mu_k\|) \quad (2.2)$$

where ω_k is the weight factor. Powell [27] showed that many properties of the interpolating function are not sensitive to the form of the non-linear function $\Phi(\cdot)$. There are many possible choices for this function. Typical choices for the radial basis functions are:

1) thin plate spline function:

$$\Phi(r) = r^2 \log(r)$$

2) multiquadric function:

$$\Phi(r) = \sqrt{r^2 + \sigma^2}$$

3) inverse multi-quadratic:

$$\Phi(r) = \frac{1}{\sqrt{r^2 + \sigma^2}}$$

4) Gaussian function:

$$\Phi(r) = \exp\left(-\frac{r^2}{\sigma^2}\right)$$

where r represents the Euclidean distance between a center and the data set, and σ is a scalar variable controlling the radius of influence of the basis function.

Broomhead and Lowe [7] pioneered the design of radial basis function based neural networks. They showed that the procedure of the RBF technique for strict interpolation is not a good strategy for the training of RBF networks because of poor generalization to new data. For applications with number of data points in the training set much larger than the number of degrees of freedom of the underlying process, the network is forced to have as many radial basis functions as the data points thus the problem is over-determined. As a result, the network may end up fitting misleading variations due to noise in the input data points, and the generalization performance is degraded significantly. Considering this problem, Broomhead and Lowe removed the restriction of strict function interpolation and designed a two-layer network structure where radial basis functions are used as computation units in the hidden layer. In their design, the training phase optimizes the fitting procedure for a desired surface based on known data points, and the generalization phases is synonymous with interpolation among data points, along the constrained surface generated by the fitting procedure.

The basic architecture of RBF networks is shown in Figure 2.1. The input data is fed into the input layer and the input layer passes it to the hidden neurons (i.e., computational units) in the hidden layer, and the output layer combines the output linearly from the hidden neurons.

Among various applicable types of radial basis functions presented previously, Gaussian function is the most common one. Gaussian function is suitable not only in generalizing a global mapping but also in refining local features without much altering the learned mapping. Gaussian function tends to be local in its response and is more biologically plausible than other functions. Further more, compared with

other types of functions such as thin plate spline function, Gaussian function is more flexible to be adjusted in terms of the function position and shape.

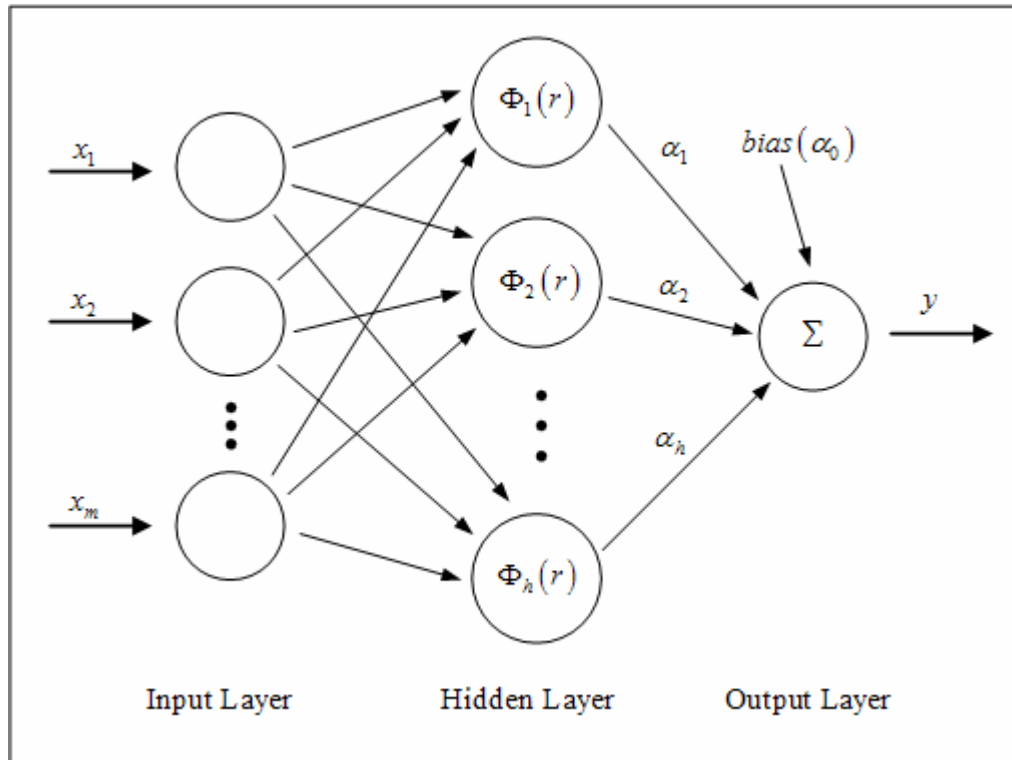


Figure 2.1: Architecture of a typical RBF network

For a Gaussian function based RBF network, four types of parameters need to be determined. They are the number of hidden units, the center positions for all hidden units, the width values for Gaussian functions, and the connection weights between the hidden units and the output nodes. Many algorithms have been designed to determine these parameters.

Poggio and Girosi [44] tried to use regularization techniques to train RBF networks. They applied the traditional regularization techniques and derived the network structures which they named as regularization networks. The regularization network has two layers and the number of hidden units is fixed a priori. A subset of the input data points are used as the centers of the hidden units. Each independent variable of

the problem is associated with a network input unit. The distance between the function represented by the neural network and the target function is measured using a cost function. They used the gradient descent algorithm to find suitable values for the weights in order to obtain accurate approximation of the underlying function. Output weights are properly initialized before the gradient algorithm is run. An approximation to the regularized solution was introduced [44] by them to decrease the high computational complexity required for finding exact solution when the sample size is very large.

Inspired by the “locally-tuned” concept of neurons from various biological nervous systems, Moody and Darken [18] constructed a network containing computational units with locally-tuned response functions. The network employs the local methods which have been used for classification, interpolation, etc. Some of computational advantages of the local methods are parallelizability and fast speed of execution. The number of hidden units is also chosen as fixed apriori. A random subset of the training data patterns are used as the hidden neuron centers. During the training, only those neurons with centers close enough to the current input pattern need to be evaluated and updated. Moody and Darken studied two types of training algorithms, a fully supervised method and a hybrid method combining the supervised method with self-organizing method. They compared the two algorithms and concluded that the hybrid algorithm worked much better and faster than the fully supervised one. In the hybrid algorithm, standard k-means clustering algorithm is used to estimate the centers of basis functions for hidden units. The width values of those basis functions are computed using “P nearest-neighbor” heuristics and the output weights are estimated by the standard Least Mean Square algorithm.

Moody and Darken [18] used the k-means clustering algorithm to estimate the centers of the basis functions. In [20], Tao pointed out that there are two potential problems associated with the k-means clustering algorithm. One is that there is still an element of chance in finding the right hidden unit centers. The other problem is

that clustering may not guarantee good results in case of function approximation because clustering is probability oriented and two input patterns close to each other do not necessarily have similar outputs. To avoid these problems, Chen et al [35] developed a procedure called Orthogonal Forward Regression to choose the RBF hidden unit centers.

All the above algorithms assume that the number of hidden units is chosen a priori. In practice, it is very difficult to find the proper number of hidden units. If the number of hidden units is set too low, the network may fail to reach a desired level of performance because of insufficient number of hidden neurons. In view of this, people are trying various approaches in the hope of recruiting new hidden units whenever necessary to improve the network performance.

Lee and Kil presented a Hierarchically Self-Organizing Learning (HSOL) algorithm for RBF in [41]. The HSOL algorithm is capable of automatically recruiting new hidden units. In HSOL, the hidden units are associated with the accommodation boundary defined in the input space and the class representation defined in the output space. The accommodation boundary of a hidden neuron is the region in the input space where it has certain influence. If an input pattern falls into a hidden unit's accommodation boundary and the input pattern and the hidden unit share the same class representation in the output space, then the network accommodates the input pattern by updating the parameters of the existing hidden units. Otherwise, a new hidden unit is created in the network. The HSOL algorithm starts from zero hidden unit and builds up hidden units gradually during the training. An alternative way to find the proper number of hidden units is to start with a large number of hidden units and remove unnecessary hidden units during the training.

So far, all the training algorithms introduced above are batch learning. Neural networks learn as the result of many presentations of a prescribed set of training examples. A complete presentation of the entire training set during the training

process is called an *epoch*. Network training algorithms can be generally categorized into two types, namely, batch learning and sequential (online) learning. In batch learning, the network parameters are updated after the presentation of all training patterns which constitute an epoch. For sequential learning, the network parameters are adjusted after the presentation of each training pattern. Several sequential learning algorithms are introduced as follows.

Platt [19] proposed a sequential learning method for a resource allocating network (RAN). In Platt's algorithm, hidden neurons are added based on the novelty of the network input data so the number of hidden units need not be selected apriori. A new input pattern is considered novel if it is far away from any of the existing center and the error between the network output and the target output is large enough. If no hidden neuron is added, the Least Mean Square (LMS) algorithm is used to update the network parameters. Kadiramanathan and Niranjana [45] improved RAN by replacing the LMS with the extended Kalman filter (EKF), and the resulting network is called RANEKF.

One of the drawbacks of the RAN and RANEKF networks is that hidden units, once allocated, can not be removed. As a result, noise data patterns may trap the network to create redundant hidden neurons. To overcome this problem, Yingwei, Sundararajan, and Saratchandran [25] improved RANEKF by adding a pruning strategy based on the relative contribution of each hidden unit to the overall network output. This improved network is known as Minimal Resource Allocation Network (MRAN). The following section gives a brief introduction to MRAN.

2.2 Minimal Resource Allocation Networks

In this section, the notations used in the MRAN algorithm are introduced first, followed by the algorithm itself. The structure of the RBF network is shown in

Figure 2.1. MRAN uses Gaussian functions as the radial basis functions for hidden units. The response of k^{th} hidden unit to the network input can be expressed as follows:

$$\Phi_k(\mathbf{x}) = \exp\left(-\frac{1}{\sigma_k^2} \|\mathbf{x} - \mu_k\|^2\right), \quad k = 1, \dots, h \quad (2.3)$$

where μ_k is the center vector of the hidden unit and σ_k is the width for the Gaussian function. $\|\cdot\|$ denotes the Euclidean norm and h represents the total number of hidden units. The output layer combines the response from the hidden layer linearly. For simplicity of illustration of the algorithm, a single output unit in the output layer is considered. Extending the algorithm to networks with multiple output units is very straightforward. Thus, the overall network response is a mapping $f: \mathfrak{R}^m \rightarrow \mathfrak{R}$, which is

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^h \alpha_k \Phi_k(\mathbf{x}) \quad (2.4)$$

where $\mathbf{x} \in \mathfrak{R}^m$ and α_0 is the bias term. The coefficient α_k is the connecting weight of the k^{th} hidden neuron to the output unit.

In this thesis, normal letters represent scalars. Bold or underlined small letters denote vectors. Bold capital letters represent matrices.

2.2.1 Introduction to MRAN

Minimal Resource Allocation Network (MRAN) is a sequential learning RBF, in which hidden neurons are added or removed as training progress. MRAN adopts the basic idea of RANEKF and improves it with a pruning strategy. The RANEKF algorithm was based on the RAN algorithm with a difference in the way of updating

network parameters. Both RAN and RANEKF algorithms have only two growth criteria and no pruning procedure. MRAN's pruning strategy enables the network to remove the hidden units which have insignificant contribution to the total output of the network, and thus the trained network becomes very compact. One of the side effects of the pruning strategy is that it may cause oscillations in the transition of the number of hidden neurons. To overcome oscillations and make the transition process smooth, an additional growth criterion based on the root mean square (RMS) error of the network output over a sliding window is introduced in MRAN. The introduction of this criterion also reduces the effect of measurement noise in the input data set. Hence, the pruning algorithm and the third growth criterion enable MRAN to obtain a compact network without compromising the accuracy of the network.

Like RAN and RANEKF, the learning process of MRAN involves the allocation of new hidden neurons as well as updating the network parameters. The MRAN network begins with no hidden units. As input data patterns are fed in, the network is built up based on certain growth criteria. For an observation (\mathbf{x}_n, y_n) , whether a new hidden unit should be added into the network depends on the satisfaction of the following three criteria:

$$\|\mathbf{x}_n - \mu_{nr}\| > \varepsilon_n \quad (2.5)$$

$$e_n = y_n - f(\mathbf{x}_n) > e_{\min} \quad (2.6)$$

$$e_{rmsn} = \sqrt{\frac{\sum_{i=n-(M-1)}^n e_i^2}{M}} > e_{\min 1} \quad (2.7)$$

where μ_{nr} is the center of the hidden unit which is closest to \mathbf{x}_n in the Euclidean sense. ε_n , e_{\min} and $e_{\min 1}$ are thresholds to be selected appropriately. The first criterion

(2.5) ensures that if a new hidden unit is to be added to the network, it must be sufficiently far away from all the existing hidden units, so that no hidden units are found to be clustered together. If the input data pattern is found to be too near to the function center of an existing hidden neuron, the first criterion is not met. In this case, no hidden unit will be added to the network and the parameters of the existing hidden units will be adjusted to accommodate this input data pattern. The second criterion (2.6) decides whether the existing hidden units are insufficient to obtain a network output that meets the error requirement. This is done through computing the output of the existing network, and comparing the network output with the target output. If the error is not significantly large, it is assumed that the existing network can be adjusted to accommodate the new input-output pair to reduce the error of the network, without the allocation of a new hidden unit.

If only the first two criteria are employed in network growth, the problem of over-fitting may come up. These two growth criteria judge the novelty of the input data solely based on the network output error and the distance between the input data and the nearest existing hidden unit center. If there is noise presented in the input data pattern, the distance between the input pattern and the nearest existing hidden neuron may be sufficiently larger than the threshold, and also the output error the network may exceed the criteria set in (2.6). In this case, a redundant hidden unit is added into the network. As the redundant hidden unit does not contribute to the network output significantly for the rest of the training process, it might be pruned later. This unnecessary allocation of a redundant hidden unit results in problem of over-fitting for the noise in the input data and a waste of computing resources.

The third criterion (2.7) is thus added to reduce the effect of noise in the training process and also to make the transition of the number of the hidden units smooth. The root mean square (RMS) value of the output error for the past M observations is calculated. If this root mean square error (RMSE), e_{rms} , does not exceed the threshold value e_{min1} , the existing hidden units are adjusted to accommodate the

new input-output pair. The calculation for the RMSE is performed over a sliding window of the size M , where the oldest network output error is eliminated to include the latest one in the averaging. A noise data may yield a large network output error and the second criterion might be met. However, if the network is already formed to approximate the input-output relationship, the network output error for the $M - 1$ previous data would be small, and also future input-output pairs would have small output errors. So a sudden ‘spike’ like increase in the output error of the network caused by the noisy data would spread over the sliding window and not exceed the RMSE threshold. As a result, no new hidden unit will be allocated for the noise data input. If the network output error is significant over a range of observations, then the RMSE value would exceed the threshold and new hidden units should be created to improve the accuracy of the network.

MRAN uses the above three criteria to effectively make the decision whether a new hidden unit should be created. When all of the three criteria are met, a new hidden unit will be added into the network with its parameters initialized as follows,

$$\alpha_{h+1} = e_n \quad (2.8)$$

$$\mu_{h+1} = \mathbf{x}_n \quad (2.9)$$

$$\sigma_{h+1} = \kappa \|x_n - \mu_{nr}\| \quad (2.10)$$

where κ is an overlap factor determining the overlap of the responses of the hidden units in the input space.

When an input data pattern to the network does not meet all of the three criteria for creating a new hidden unit, the network parameters are adjusted using the extended Kalman filter (EKF) algorithm. The RAN algorithm uses least means square (LMS) algorithm. MRAN employs EKF instead of LMS because Kadirkamanathan &

Niranjan showed that RANEKF exhibits superior performance over the original RAN [45].

For MRAN, when the observation (\mathbf{x}_n, y_n) does not meet all the criteria for adding a new hidden unit, the network parameters $\mathbf{w} = [\alpha_0, \alpha_1, \mu_1^T, \sigma^1, \mathbf{L}, \alpha_h, \mu_h^T, \sigma^h]^T$ are adapted using the extended Kalman filter. The EKF algorithm computes the posterior estimate \mathbf{w}_n from its prior estimate \mathbf{w}_{n-1} as follows:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{k}_n e_n \quad (2.11)$$

where e_n is the network output error and \mathbf{k}_n is the Kalman gain vector given by,

$$\mathbf{k}_n = \mathbf{P}_{n-1} \mathbf{a}_n \left[\mathbf{R}_n + \mathbf{a}_n^T \mathbf{P}_{n-1} \mathbf{a}_n \right]^{-1} \quad (2.12)$$

where \mathbf{a}_n is the gradient vector with the following form:

$$\mathbf{a}_n = \nabla_{\mathbf{w}} f(\mathbf{x}_n) = \begin{bmatrix} 1, \Phi_1(\mathbf{x}_n), \Phi_1(\mathbf{x}_n) \frac{2\alpha_1}{\sigma_1^2} (\mathbf{x}_n - \mu_1)^T, \\ \Phi_1(\mathbf{x}_n) \frac{2\alpha_1}{\sigma_1^3} \|\mathbf{x}_n - \mu_1\|^2 \\ , \mathbf{L}, \\ \Phi_h(\mathbf{x}_n), \Phi_h(\mathbf{x}_n) \frac{2\alpha_h}{\sigma_h^2} (\mathbf{x}_n - \mu_h)^T, \\ \Phi_h(\mathbf{x}_n) \frac{2\alpha_h}{\sigma_h^3} \|\mathbf{x}_n - \mu_h\|^2 \end{bmatrix}^T \quad (2.13)$$

\mathbf{R}_n is the variance of measurement noise. The error covariance matrix \mathbf{P}_n is computed by

$$\mathbf{P}_n = \left[\mathbf{I} - \mathbf{k}_n \mathbf{a}_n^T \right] \mathbf{P}_{n-1} \quad (2.14)$$

where \mathbf{I} is an identity matrix.

Because of the rapid convergence of the EKF algorithm, it may prevent the model from adapting to future data. To solve this problem, a random walk model was proposed in [45] to add to the above equation, which now becomes

$$\mathbf{P}_n = [\mathbf{I} - \mathbf{k}_n \mathbf{a}_n^T] \mathbf{P}_{n-1} + Q\mathbf{I} \quad (2.15)$$

where Q is a scalar that determines the allowed random step in the direction of gradient vector. \mathbf{P}_n is a positive definite symmetric matrix. When a new hidden neuron is allocated, the dimensionality of \mathbf{P}_n increases and thus, the new rows and columns must be initialized. Because \mathbf{P}_n is an estimate of the error covariance of the parameters, \mathbf{P}_n is chosen as

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_n & 0 \\ 0 & P_0 \mathbf{I} \end{pmatrix} \quad (2.16)$$

where P_0 is an estimate of the uncertainty in the initial values assigned to the parameters. The dimension of the identity matrix \mathbf{I} is equal to the number of new parameters introduced by the allocation of the new hidden unit.

In MRAN, the distance threshold of the input data pattern to the nearest hidden unit ε is decayed gradually during the training process. ε begins with $\varepsilon = \varepsilon_{\max}$, the largest scale of interest, typically the size of the entire input space of nonzero probability. The distance threshold at the n^{th} observation ε_n is decayed exponentially as follows,

$$\varepsilon_n = \max \{ \varepsilon_{\max} \gamma^n, \varepsilon_{\min} \} \quad (2.17)$$

where $0 < \gamma < 1$ is the decay constant. The value of ε is decayed until it reaches ε_{\min} . The exponential decaying of the distance threshold allows fewer basis functions with large widths (smoother basis functions) initially and with increasing number of observations, more basis functions with smaller widths are allocated to fine tune the approximation.

MRAN's pruning strategy was inspired by [48], in which Cheng proposed a method of pruning hidden neurons during the network batch training. In MRAN, the pruning operation is performed in a sequential way. In [48], only the weight value for each hidden neuron is considered, while MRAN takes the whole output value of each hidden unit into consideration. The contributed output made by the k^{th} hidden unit to the output layer is expressed as follows,

$$o_k = \alpha_k \exp\left(-\frac{1}{\sigma_k^2} \|\mathbf{x}_i - \mu_k\|^2\right) \quad (2.18)$$

It can be seen from the expression above that the contribution of a hidden unit depends on three key values, the weight α_k , the width σ_k and the Euclidean distance $\|\mathbf{x}_i - \mu_k\|$. If the weight value is very small, the output contribution of the hidden unit will be small. If the width value is small, the contribution will rapidly decrease from its centre, μ_k . If the distance between the input data pattern and the hidden unit is large, the output contribution will be small too.

The k^{th} hidden unit is eligible to be pruned only if its output contribution to the network output is insignificant over a time period whose length is predefined. Since the output contributions of hidden units vary as different input data patterns are fed into the network, the comparison is done with normalized values instead of the absolute values. The normalization is done through dividing the output of all the

hidden units by the value of the highest output among the units. A normalized output contribution value r_k is calculated as follows,

$$r_k = \frac{\|o_k\|}{\|o_{\max}\|} \quad (2.19)$$

where $\|o_{\max}\|$ is the largest absolute hidden unit output value. If the normalized output contribution of a hidden unit is less than a threshold δ , for n_w consecutive observations, then it contributes insignificantly to the output layer and should be pruned from the network.

The pruning strategy prunes not just hidden units that do not contribute to the output significantly, but also those that are too close to one another. The former is done by observing the output of each of the hidden units for a number of consecutive observations, and then pruning the hidden units that have not been contributing to the network output significantly. The latter is performed if the distance between two hidden neurons are found to be less than a threshold value, and the difference between their width parameter is less than a threshold value. The two hidden neurons are then combined to form a single hidden unit. When pruning takes place, the dimensions of the EKF are reduced accordingly to suit the resized network.

The pruning strategy effectively prunes hidden units that may have been active initially, but subsequently end up contributing little to the network output. It also ensures that hidden neurons do not form clusters that are too close to each other. The detection and removal of such hidden units make it possible to result a compact network.

One of the advantages of MRAN algorithm over other sequential learning algorithms is that the hidden neuron centers are not fixed apriori. Thus the user does not have to care about choosing the right number of hidden units. The pruning

strategy also makes it possible to achieve a very compact network. The following subsection gives a brief description of successful applications of MRAN.

2.2.2 MRAN Applications

MRAN has been successfully applied to various areas such as function approximation, pattern classification, communication channel equalization, etc.

In [26], MRAN was used to perform function approximation on a static function consisting of the summation of six exponential functions. Compared with RANEKF, MRAN gives a very compact network.

MRAN has also been applied to some benchmark problems and its performance is compared with other learning schemes. In [45], Kadiramanathan and Niranjana employed both RAN and RANEKF for function approximation on Hermite polynomial function. MRAN has also been used to perform Hermite polynomial function approximation. It has been shown that MRAN achieved the smallest approximation error with the most compact network, when compared with RAN and RANEKF. PROBEN1 is a collection of benchmark problems for neural network learning in the realm of pattern classification and function approximation. It also offers a set of rules and conventions for carrying out benchmark testing. These benchmark data sets are available online at http://www.wipd.ira.uka.de/~prechelt/NIPS_bench.html. There are fifteen data sets from different domains. MRAN has been used to solve two function approximation problems (*hearta* and *flare*) and one pattern classification problem (*cancer*). For the *hearta* problem, it was found that MRAN produced a smaller approximation error with a more compact network than multi-layered feed-forward network (MFN). The magnitude of the number of free network parameters used by MRAN was also one or two orders less than that of the MFN. MRAN produced the similar

approximation error on the *flare* problem as the MFN, but achieved a much smaller network.

MRAN has also been used in nonlinear dynamic systems. MRAN has been applied in chaotic time series prediction. In particular, the Mackey-Glass time series were used as the benchmark problem. The output error of MRAN was very similar to that of RAN and RANEKF, however, the network trained by MRAN was much smaller than those of RAN and RANEKF in terms of number of hidden units. MRAN has also exhibited advantages in fields like currency exchange rate prediction, non-linear signal processing, non-linear system identification, etc.

Additionally, MRAN has been used for channel equalization. It was found that MRAN achieves comparable performance to the optimal Bayesian decision equalizer and MRAN is much faster than the optimal Bayesian equalizers in terms of CPU time [31].

With many successfully applications of the MRAN algorithm in various areas, it is natural that MRAN's application on speaker verification should be investigated.

Chapter 3

Review of Speaker Verification

3.1 Introduction to Speaker Verification

Identity verification becomes pervasive in our daily life. For instance, Automatic Teller Machines (ATMs) ask the user to input a password after the user inserts his or her bank ATM card for the purpose of identity verification. Access controlling systems to buildings are another example of identity verification systems.

Although the above verification mechanisms may be quite effective, they have a big drawback – the systems become vulnerable to stolen ATM cards and compromised passwords. In view of this, researchers are attempting to develop new verification methods based on biometric such as human being's voice, fingerprints, retinas because these physical attributes are rarely changed and vary significantly from person to person.

Speech contains many unique characteristics that are specific to each individual. Many of these characteristics are independent of the linguistic message of the

speech. From the speech signal alone good guesses can be made as to whether the speaker is female or male, child or adult. A careful analysis of the speech signal can reveal many more characteristics of the speaking person.

If speech samples from a group of people are collected, processed and stored in a computer system, it is possible for the computer to recognize members of the group by their voices at a later time. The computer can also detect imposters, i.e., people falsely claiming to be members of the group. If an unknown speaker claims to be a certain person, the computer can verify the claim based on a comparison of their respective speech signals. These are some examples of speaker recognition.

While the area of speech recognition is concerned with extracting the linguistic message underlying the speech, speaker recognition is concerned with extracting the identity of the person speaking it. Speaker recognition is the process of automatically recognizing the identity of the person who is speaking by using the speaker's characteristic information included in speech waves [36][37].

Speaker recognition is the general term used to include all of the many different tasks of discriminating people based on their voices. The area of speaker recognition can be classified into speaker identification and speaker verification depending up the nature of the application.

Speaker identification is the process of determining which one of a group of known voices best matches the input voice sample. It is a N -class decision task, where N is the number of candidate speakers. The value of N has a significant influence on the performance for a speaker identification task. The performance of a speaker identification system may be very good when there are only a few speakers. However, the performance could be very bad when N becomes arbitrarily large. The speaker identification task is referred to as "closed-set" task when the actual speaker is always one of the enrolled speakers. In most cases, the speaker identification task appears to be less of an operationally useful capability than a

scientific game [10]. Though the speaker identification task may attract some scientific interest, it's speaker verification that has great application potential.

Speaker verification is the process of accepting or rejected the identity claim of a speaker from a voice sample. It is a 2-class decision task. The two classes are: the target speaker and some speaker other than the target speaker (also known as an imposter). As mentioned, the performance of a speaker identity task varies as the number of candidate speakers changes. However, the performance of a speaker verification task is independent of the number of potential imposter speakers. The speaker verification task is also referred to as the “open-set” task, because it distinguishes a claimed speaker’s voice known to the system from a potentially large group of voices unknown to the system (i.e., imposter speakers).

Depending on the operating mode of the applications, speaker recognition can be divided into text-dependent and text-independent.

If the speaker recognition system is based on voice recording of the same utterance in both the training and operating phases, i.e., the system knows what the speaker is supposed to say, it is text-dependent. In a text-dependent speaker recognition system, the speaker will either say a predefined utterance or will be prompted to say an utterance by the system. In both cases, the target speaker is required to explicitly speak these utterances during the training phase. Text-dependent speaker recognition generally achieves higher recognition performance than the text-independent method because the explicit knowledge of the utterance can be used to build detailed dynamic models of the speaker with great accuracy. However, for many applications, like forensic and surveillance applications [12], predetermined utterance can not be used. In this case, text-independent speaker recognition should be used.

If a speaker recognition system does not have foreknowledge of the utterance spoken by the speaker, it is called “text-independent”. A text independent system

allows any utterances to be used in both training and operating phases. The system is capable to use whatever speech data happens to be available. Text-independent systems often exhibit worse performance than text-dependent systems do, due to lack of the explicit knowledge of the utterance. However, the general absence of idiosyncrasies in the definition of the task and the specification of the speech data makes it possible for text-independent systems to support much broader collaboration and sharing of results than has been realized in text-dependent recognition. As a result of this signification advantage, interest in text-independent speaker recognition has risen boomingly.

This thesis focuses on text-independent speaker verification.

3.2 Acoustic Parameters

Before performing speaker verification, we need to identify and derive measurable parameters (features) of recognition. In a speaker recognition system, the computer creates and stores models of known speakers' voices based on parametric measurements. When speech form an unknown source is presented, the system creates a parametric model and compares it with existing ones. In the case of speaker verification, the model created from the speech sample is compared with the model for the target speaker. If the models are sufficiently similar, the verification is made.

A microphone is used to collect speech waves from the speaker and generate the corresponding electrical signals. The electrical output of the microphone is an analog signal, which varies smoothly with time. Since computers have problems to store analog signals, the signal is transformed into a digital signal through an analog-to-digital (A-to-D) converter. The A-to-D converter reduces the continuously varying voltages of the electrical signal to a sequence of digits through

proper sampling. A quantization process rounds the digits into the nearest representable values for the computer. Once the speech signal is sampled and quantized, it is ready for parameter extraction.

3.2.1 Pitch

Pitch is the simplest acoustic parameter. There are several ways to perform pitch extraction. For example, it can be determined by counting the glottal pulses (vocal cord vibrations) in one second in a spectrogram. Another way is to compute it from the autocorrelation function. The autocorrelation function compares pieces of the signal in the time domain with one another. When values they correlate (i.e., values tend to be the same at different times), the time difference is a multiple of the pitch period, and the shortest one that produces a high correlation is the actual pitch period. There are several other methods for measuring the pitch period in [24].

The average pitch over an utterance is useful at a rough level of speaker recognition. It can be used to distinguish among male, female speakers. However, it is incapable of distinguishing among people whose voices are at similar pitch levels. Another drawback of pitch is that the pitch level may be affected by the speaker's mood. In practice, more advanced parameters should be used instead of pitch.

3.2.2 Frequency

Speech signals can be represented in the frequency domain. A speech spectrogram is one such representation, which is usually computed from a discrete Fourier transform. It consists of a sequence of triples of numbers, time, frequency, and amplitude. Collectively, these numbers contain characteristic information about the speaker such as formant frequencies. Formant frequencies are related to the shape of the individual's resonant cavities – one of many characteristic anatomical features

of the vocal tract. At first, researchers attempted to use spectrograms pictorially to determine speaker similarity, often through coarse features such as formant widths and shapes. The computer system can be used to perform more precise calculations on the raw numbers underlying the spectrogram, thus to achieve more accurate results. However, frequency is still not a good candidate to perform highly accurate speaker recognition.

3.2.3 Linear Predictive Coefficients

Linear predictive coefficients are widely adopted as the most effective speech features. It has been used in various areas of speech processing, such as speech synthesis, speech recognition, and speaker recognition. It was found that linear predictive coefficients are the best features for speaker recognition [3].

Linear predictive coding (LPC) is the most widely used vocoder. LPC makes use of the fact that speech signals change relatively slowly most of the time. For instance, a vowel may last for several hundred milliseconds, but fricative durations are around 100 ms. These are short intervals for human view but a rather long time for computer systems. In LPC, a spectral parameter, a loudness parameter, and certain parameters related to the vocal tract shape are extracted from the speech signal as infrequently as every 30 ms (the frame size) and the characteristics of the speech signals are still persisted accurately. Methods to compute LPC coefficients can be found in [34].

In noisy environments, delta-cepstrum coefficients are calculated by computing the differences between cepstral coefficients in each time frame. Constant bias caused by noise or channel distortion is then removed. Relative spectral-based coefficients (RASTA) utilize a series of transformations to eliminate signal distortion. Stationary and slow-moving variations in the frequency domain are detected and

removed. Fast-moving variations, which represent the speech, are captured in the resulting parameters.

There are many other kinds of parameters that have been used at various times. However, in practice, LPC derived coefficients are the most suitable parameters for building speaker models.

3.3 Speaker Verification Methods

Once acoustic parameters are obtained, they can be incorporated in a model representing the speaker. The model should be constructed in a way allowing the computation of distance measure between two models to reflect the degree of difference between two corresponding speakers. This section presents several modeling techniques.

3.3.1 Long-Term Statistical Methods

Long-term statistical methods use long-term sample statistics of various spectral features, such as the mean and variance of spectral features over a series of utterances [38].

Pitch or cepstral coefficients are often used as the acoustic parameters in long-term statistical methods. Long-term statistical methods require a large number of feature vectors collected from each known speaker. The average and variance of each component of the feature vector are computed and each speaker is modeled by a vector of average values and a vector of variances. The unknown speaker is modeled with a similar model.

There exist many techniques to measure the distance between these computed vectors. Euclidean distance is the most commonly used measure, which considers each vector to represent a point in a multidimensional space and computes the distance between the points. In Euclidean distance measure, more significance is given to dimensions that have low variance, because low variance indicates consistency and as a result such measurements are often quite reliable. Long-term statistical methods are useful when large amounts of data are available to construct speaker models. Such methods are not effective if utterance are short because there will not be enough data to be statistically significant.

Long-term statistical methods do not perform well because these statistics are a minimal representation of spectral characteristics and can also be sensitive to the variability of the transfer function of the transmission channel. More recently, Montacie et al. improved statistical methods by using a multivariate autoregressive model to model the statistics of dynamic variables in the spectral domain [6]. Multivariate autoregressive (MAR) model is a classical tool for processing signals with various components. The multivariate linear prediction describes the spectral evolution of speech signals using a MAR model excited by a white vectorial noise. For any N spectral vectors $\{y_n\} (n=1, L, N)$ of dimension p , their evolution can be represented by a MAR model of order q ,

$$y_n = -\sum_{i=1}^q A_i y_{n-i} + e_n \quad (3.1)$$

where A_i is a $p \times p$ matrix and e_n represents a white noise. Levinson-Whittle-Robinson algorithm [33] is used to estimate the coefficients of the matrix A_i .

The MAR model has been applied to speaker recognition [8][42]. The main difficulty for MAR modeling is to properly estimate the optimal order q . In [6], it was found that order 2 MAR model is the best choice. Their experiments showed

that for a MAR model order over 2, the prediction error does not decrease significantly but the recognition rates decrease considerably. [5] further confirmed that the optimal order of the MAR model is around 2 or 3. It was also found that correct normalization of scores according to a posteriori criteria is essential to good performance in [5].

3.3.2 Vector Quantization

Short-term feature vectors extracted from speech samples of a speaker can be used directly to represent the essential characteristics of the speaker. However, when the number of training vectors becomes very large, such a direct representation is not possible because the memory and amount of computation required will be prohibitively large. In this case, vector quantization techniques can be used to compress the training vectors [49].

By applying vector quantization, the spectral characteristics of each speaker can be modeled by one or more codebook entries that are representative of that speaker.

The speaker based vector quantization codebook generation can be described as follows:

For a set of I training feature vectors, $\{a_1, a_2, \dots, a_I\}$, which characterize the variability of a speaker, we need to find a partitioning of the feature vector space, $\{S_1, S_2, \dots, S_M\}$, for the speaker where S , the whole feature space is represented as $S = S_1 \cup S_2 \cup \dots \cup S_M$. Each partition in the feature space, S_i , forms a convex, nonoverlapping space and every feature vector inside S_i is represented by the corresponding centroid vector \mathbf{b}_i . The partitioning is performed in such a way to minimize the average distortion over the whole training set. The average distortion is calculated as follows:

$$D = \frac{1}{I} \sum_{i=1}^I \min_{1 \leq j \leq M} d(\mathbf{a}_i, \mathbf{b}_j) \quad (3.2)$$

where $d(\mathbf{a}_i, \mathbf{b}_j)$ is the distortion distance between the feature vector \mathbf{a}_i and the corresponding feature vector space partition's centroid vector \mathbf{b}_j .

As mentioned, linear predictive coefficients (LPC) are the most commonly used feature vectors. For LPC vectors, the distortion distance measuring the similarity between any two feature vectors is the LPC likelihood ratio distortion measure. The value of the likelihood ratio distortion between two LPC vectors \mathbf{a} and \mathbf{b} can be computed as follows:

$$d_{LR}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{b}^T \mathbf{R}_a \mathbf{b}}{\mathbf{a}^T \mathbf{R}_a \mathbf{a}} - 1 \quad (3.3)$$

where \mathbf{R}_a is the autocorrelation matrix of input data associated with the vector \mathbf{a} . The speaker based vector quantization codebooks can be generated using this distortion measure the vector quantization algorithm proposed in [9].

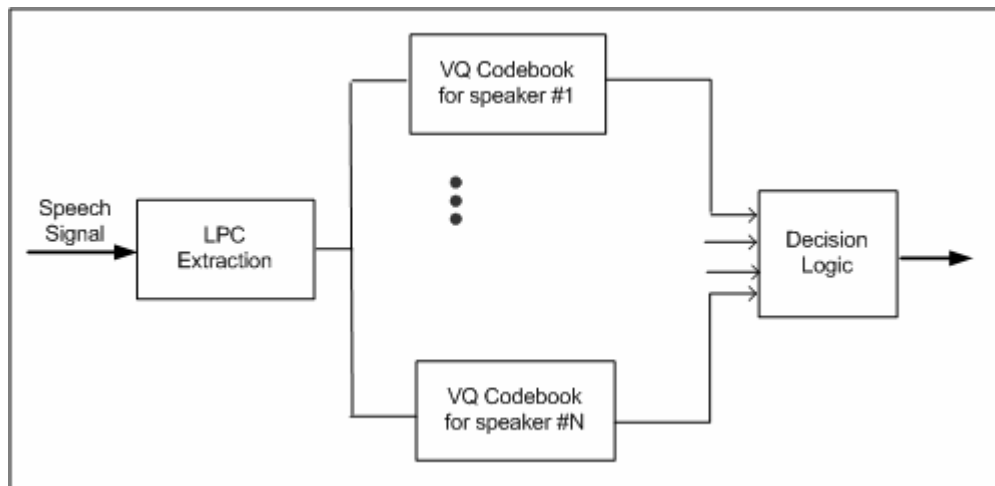


Figure 3.1: Speaker-based vector quantization system

The speaker recognition system based on this vector quantization codebook method can be illustrated in Figure 3.1.

The input signal is sampled, and LPC gives the sequence of vectors $\{a_1, a_2, \dots, a_L\}$. The LPC vectors are then vector quantized using N codebooks corresponding to the N candidate speakers. The quantization distortions for each codebook are separately accumulated across the whole training token. The average quantization distortion for the i^{th} codebook is:

$$D_i = \frac{1}{L} \sum_{l=1}^L \min_{1 \leq j \leq M} d(\mathbf{a}_l, \mathbf{b}_j^i) \quad (3.4)$$

The minimum value of the N average quantization distortions can be found easily. The final speaker recognition decision is then given by:

$$i^* = \arg \min_{1 \leq i \leq N} D_i \quad (3.5)$$

The speaker verification decision can be made through a speaker-verification system with a similar structure consisting of only the codebooks of the claimed identity. The resultant average distortion is compared with a predefined threshold to accept or reject the identity claim made by the unknown speaker.

It was found that large codebook size and long training token length can be used to improve the speaker recognition performance. It was also recommended that the vector quantization codebook should be updated continuously to alleviate the performance degradation due to different speech signal collection conditions and intra-speaker variations.

The vector quantization approach introduces a new factor affecting the performance – the codebook size. When a codebook consists of only one vector, vector quantization is the same as long-term statistical averaging method. Although larger

codebooks perform better in characterizing a speaker's voice, the computational expense increases and the as a result, more time is required. On the other hand, vector quantization is valuable because the amount of data in the speaker's model is drastically reduced without much loss in accuracy.

3.3.3 Hidden Markov Models

Hidden Markov Models (HMMs) can also be used to model speakers. HMMs are particularly useful for modeling the stationary and transient properties of speech signals. In this approach, a fully connected (ergodic) HMM is trained during the enrollment phase for each speaker. The states of the HMM are then defined in a completely arbitrary and unsupervised manner. In this approach, distances are stochastic and trained. Alternatively, states can be associated with specific classes. Temporal constraints can be included in the models through introducing minimum duration constraints on each state. Finally, a probability density function is associated with each state. Savic and Gupta [28] employed an autoregressive process to estimate the probability distribution associated with each state.

Spectral envelope parameters are important features for speaker recognition. The spectral envelope can be represented by short-term autocorrelation coefficients of speech signals. For a short period of time, the speech samples can be effectively modeled by an autoregressive source, with either periodic or random excitation [4]. The short-time spectral envelope of the speech signal can be represented by a small number of autoregressive parameters. However, for longer time periods, the autoregressive parameters fall into well defined clusters (states).

These well defined states or clusters can be modeled in a hidden Markov model, and the temporal variation in the parameters can be represented by Markovian transitions between states. A linear predictive HMM is obtained by modeling signal in each state as an autoregressive source. The resulting Markov model is a hidden

model because an autoregressive model with Gaussian excitation is a stochastic process. For more details on hidden Markov modeling, please refer to [4][23].

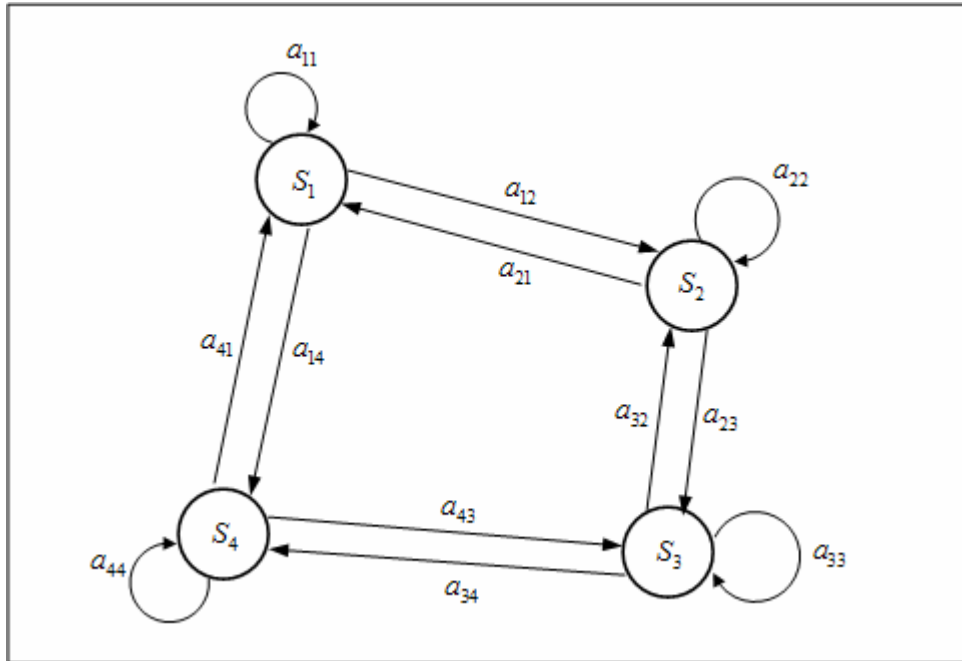


Figure 3.2: A four-state HMM

A hidden Markov model consists of two components: a Markov process and a set of stochastic functions or output probabilities. Assume that the underlying Markov chain has N states. (A four state ergodic HMM is illustrated in Figure 3.2) The components of a HMM are defined as follows:

- 1). Initial probabilities: $\pi_j = P[q_1 = s_j], 1 \leq j \leq N$, where q_1 denotes the actual initial state and s_j is the j^{th} state node.
- 2). Transition probabilities are determined by the matrix $A = [a_{ij}]$, where $a_{ij} = P[q_{t+1} = s_j | q_t = s_i], 1 \leq i, j \leq N$. q_t represents the actual state at time t , and $S = \{s_1, s_2, \dots, s_N\}$ represents the N states.

3). Output probabilities $b_j(o_t) = P[o_t = s_j], 1 \leq j \leq N$.

$o_t = (x_0, x_1, \dots, x_{K-1}) \in \mathcal{R}^K$ is an observation vector (a frame of speech) normalized by the average residual energy per sample. Let $\mathbf{a} = (a_0, a_1, \dots, a_p)$ be the corresponding autoregressive vector, then:

$$b_j(o_t | \mathbf{a}) = (\sqrt{2\pi})^{-K} \exp\left\{-\frac{\delta(o_t; \mathbf{a})}{2}\right\} \quad (3.6)$$

where

$$\delta(o_t; \mathbf{a}) = r_a(0)r(0) + 2\sum_{i=1}^p r_a(i)r(i) \quad (3.7)$$

$$r(i) = \sum_{n=0}^{K-i-1} x_n x_{n+i} \quad (3.8)$$

$$r_a(i) = \sum_{n=0}^{p-i} a_n a_{n+i}, \quad (a_0 = 1), 0 \leq i \leq p \quad (3.9)$$

A HMM model λ is denoted by $(\mathbf{A}, \mathbf{B}, \pi)$. With a hidden Markov model and an observation sequence $\mathbf{O} = (o_1, o_2, \dots, o_T)$, the total observation probability can be computed using the Forward-Backward algorithm. Additionally, the Baum-Welch algorithm provides an iterative solution to the estimation of HMM parameters.

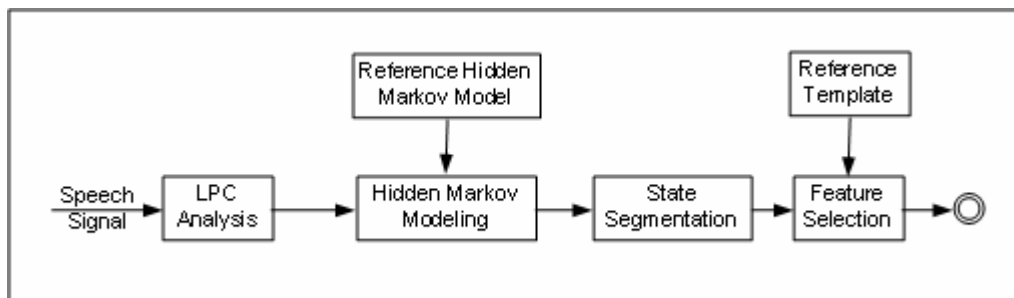


Figure 3.3: Flow chart diagram for the training phase

During the enrollment phase, a hidden Markov model is obtained for each speaker. The training process is illustrated in Figure 3.3. In a linear predictive HMM, the observation probability density function for a state is represented by an autoregressive vector. States representing the desired broad phonetic categories can be identified through spectral analysis of the LPC vectors. A feature selection is performed after identifying frames belonging to particular classes of phonemes. In this step, reference templates are constructed and verification threshold is computed. The verification threshold consists of the Mahalanobis distance and the number of verification votes required to make an accept decision.

In the verification phase, different classes of phonemes are obtained using the reference HMM. For each phonetic class, a test feature template is computed using the Hotelling trace algorithm. A verification score can be obtained by comparing the test feature template with the corresponding reference template. The same procedure is thus repeated for each class. A weighted linear combination of the all the scores forms the final verification score. Thus, the verification decision can be made based on the score.

Tishy [32] generalized the autoregressive HMM to the class HMMs using mixtures of autoregressive processes. Besides autoregressive HMMs, there are other types of HMMs for speaker verification. Matsui [43] trained HMMs according to a maximum likelihood criterion. Some discriminant training methods have also been used in speaker verification to improve discrimination between speakers [22].

3.3.4 Artificial Neural Networks

Oglesby and Mason [16][17] applied multilayer perceptrons on speaker recognition. They modeled each speaker using a personalized feed-forward neural network.

The differences in people's speech are directly modeled by including speech from many people in the training data of each net. A neural model is constructed for each candidate speaker, and trained with speech from the representative sample of the speakers known to the system. The neural network then learns to have an active response for speech associated with a single given speaker and an inactive response for the speech of other people. The network is trained by minimizing the mean squared error between the desired output and the actual output of the network through a conjugate gradient descent algorithm. For speaker identification, each neural network is trained to the same level of accuracy in terms of mean squared error on the training data, so that the outputs of each one represent a balanced view of the likelihood that the test utterance comes from a given speaker. In the enrollment phase, the amount of training data from speakers other than the one associated with a particular network is much larger than that from the speaker. This results a local minimum with a large basin of attraction in the training error function, and networks so trained would be inactive for all persons' speech. To avoid this problem, the model is weighted to balance the amount of active and inactive training patterns.

One of the advantages of this approach is that the neural model does not need to grow in size proportional to the number of speakers known to the system. This holds despite modeling of differences from potentially a large population. Restricting the model size is crucial for acceptable training time and the quality of solution. Since the training of models represents a very difficult non-linear optimization problem, it tends to produce less optimum solutions as the larger and more difficult the problem becomes. To reduce the training time, the training data for modeling speech differences are drawn from only a proper partition of the whole population.

Ogleby and Mason [16] found that large neural models with two hidden layers are inferior to models with only a single hidden layer and fewer weights. They

compared the results using neural nets with those by other methods. The recognition performance for neural networks is comparable to that of a vector quantization approach. The neural method outperforms the codebook system for small model sizes but performs worse for larger models.

More recently, Mak and Kung [29] applied both radial basis function (RBF) networks and elliptical basis function (EBF) networks to speaker verification. Their experimental results showed that small EBF networks with basis function parameters estimated by the expectation-maximization (EM) algorithm outperform large RBF networks. It has also been shown that the equal error rate achieved by their EBF networks is less than that achieved by the vector quantization based models.

3.4 Database for Speaker Verification – TIMIT

In order to evaluate and compare various speaker verification algorithms, it is highly desirable to have a carefully designed acoustic-phonetic corpus. The Texas Instruments/Massachusetts Institute of Technology (TIMIT) acoustic-phonetic corpus is one of the most widely used speech databases for speaker recognition [47].

TIMIT contains a total of 6300 utterances – 10 sentences spoken by each of 630 speakers from eight major dialect regions of the United States. The 10 sentences represent around 30 seconds of speech per speaker. 70 percent of the speakers are male and 30 percent are female. All speakers are native speakers of American English and none of them has any clinical speech pathology.

These 630 speakers were selected to represent different geographical dialect regions of the U.S. The dialect region of a speaker was defined as the geographical area of the U.S. where he or she lived during their childhood years (age 2 to 10).

The detailed dialect distribution of speakers is shown in Table 3.1.

Table 3.1: Dialect distribution of speakers in TIMIT

DIALECT REGION		MALE SPEAKERS	FEMALE SPEAKERS	TOTAL
Name	No.			
New England	1	31	18	49
Northern	2	71	31	102
North Midland	3	79	23	102
South Midland	4	69	31	100
Southern	5	62	36	98
New York City	6	30	16	46
Western	7	74	26	100
Army Brat (moved around)	8	22	11	33
TOTAL		438	192	630

Speech recordings were made in a noise-isolated recording booth, using a semi-automatic computer system to control the presentation of prompts to the speaker and the recording. The speech was directly digitalized at a sample rate of 20 kHz with the anti-aliasing filter at 10 kHz. The speech was then digitally filtered, debiased and downsampled to 16 kHz.

The speakers were seated in the recording booth and prompts were presented on a monitor. The speakers were given minimal instructions and asked to read the prompts in a natural voice. Any suspected mispronunciations were flagged for verification. When a pronunciation error was detected, the sentence was re-recorded. The text material in the TIMIT prompts consists of two dialect “shibboleth” sentences designed at Stanford Research Institute (SRI), 450 phonetically-compact sentences designed at MIT, and 1890 phonetically-diverse sentences selected at Taxes Instruments. Table 3.2 summarizes the speech material in TIMIT.

The dialect sentences (the SA sentences) were meant to expose dialectal variants of the speakers and were read by all speakers. The phonetically-compact sentences (the SX sentences) were hand-designed to be comprehensive as well as compact. Each speaker read three SX sentences. The phonetically-diverse sentences (the SI sentences) were selected from some existing text sources in order to add diversity in sentence types and phonetic contexts.

Table 3.2: TIMIT speech material

SENTENCE TYPE	NO. OF SENTENCES PER SPEAKER
Dialect (SA)	2
Compact (SX)	5
Diverse (SI)	3
Total	10

In this thesis, the TIMIT corpus is used as the test base to evaluate our algorithms and then compare them with others.

Chapter 4

MRAN for Speaker Verification

Many approaches have been proposed to perform speaker verification. In this chapter, we will apply MRAN to speaker verification and compare the results of MRAN with those of other methods such as traditional radial basis networks and elliptical radial basis networks. The experimental results show that MRAN produces comparable error rates to other methods but with much less computational complexity.

4.1 Overview

Speaker verification is the process of accepting or rejecting the identity claim of a speaker from a voice sample. In order to achieve good performance for speaker verification, we need to model the speaker properly first.

The previous chapter presented many different modeling techniques to model the speaker based on utterances of the true speaker. In our experiments, we use a

MRAN network to model a speaker. Various phases involved in the experiments are illustrated in Figure 4.1.

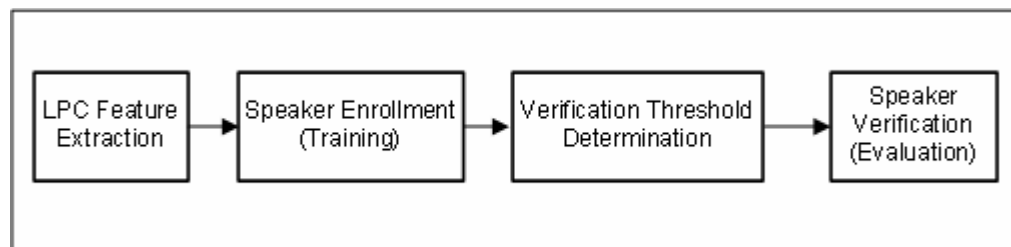


Figure 4.1: Phases involved in the experiments

During the training process, the MRAN network builds up as the utterances are fed in. Speech signals are first processed through a LPC analysis system. Linear predictive (LP) derived cepstral coefficients are obtained from the LPC system. These LP-derived cepstral coefficients are used as feature vectors. The number of input units in the MRAN network is equal to the number of elements in a feature vector. The number of network output unit is one. The training of the MRAN network for the speaker is supervised. Feature vectors of voice samples from the speaker and other speakers (imposters) are used as the input. For feature vectors extracted from the speech samples of the speaker, the output is set to be active, and for feature vectors from other speakers, the output is set to be inactive.

After the MRAN network for the speaker has been properly trained, the verification threshold for the network needs to be determined. The verification threshold is used to make the decision of accepting or rejecting the identity based on the output (or the average output value of a certain number of outputs) of the network in operating mode. If the output value is larger than the verification threshold, the decision is accepting the identity claim; otherwise, the claim is rejected. The verification threshold is set to the value at which the network's false accept rate (the probability of falsely detecting the target speaker when not present) is reasonably small (e.g.,

2%). More details on determination of verification threshold can be found in section 4.5.

After the network has been trained and the verification threshold has been determined, the network is now capable of performing speech verification based on the input speech signals. There are several metrics that are used to evaluate the performance of a speaker model, such as false accept rate, false reject rate, and equal error rate. By comparing these rates of MRAN and those of other approaches, we can compare the performance of MRAN with that of other methods. Computational complexity of algorithms should also be considered in the comparison.

This chapter presents our experiments of application of MRAN on speaker verification tasks. The TIMIT database is used as the test database. Our experimental results are compared with those from others.

The sequential learning algorithm of MRAN used in the experiments can be summarized as follows:

- 1) For each observation, compute the overall network output;
- 2) Calculate the parameters required in the growth criterion, such as output error, the distance between the input vector and the nearest hidden unit;
- 3) IF all of the following three conditions are satisfied:
 - I. The output error exceeds a minimum threshold value;
 - II. The root mean squared error of the network over a window with certain size is above a certain threshold value
 - III. The distance between the input vector and its nearest hidden unit is sufficiently large
- 4) THEN Create a new hidden unit;

- 5) ELSE Adjusts the weights and widths of the existing hidden neurons by using the extended Kalman filter (EKF) algorithm;
- 6) IF a hidden unit's normalized contribution to the output for a certain number of consecutive inputs is below a threshold, this hidden unit is pruned.
- 7) IF two hidden units are found to be close to each other, the two hidden units are combined into a signal unit;
- 8) Adjusts the dimensions of EKF if hidden units are added, pruned or combined;
- 9) Repeat the steps 1-8.

4.2 Speech Database and Feature Extraction

All our experiments use the TIMIT corpus introduced in last chapter. TIMIT is a phonetically balanced continuous speech corpus, which contains speech from 630 speakers representing 8 major dialect divisions (regions) of American English.

In our experiments, 258 speakers (157 male and 101 female) from the first four regions of the corpus are used in the experiments. These speakers are divided into four sets: the speaker set, the anti-speaker set, the pseudo-imposter set, and the imposter set. The details on speakers in each set are shown in Table 4.1. The purposes of each set are briefly described below.

The speaker set contains all the candidate speakers. In fact, one candidate speaker is enough to perform speaker verification. However, the result of such an experiment will be very speaker-dependent. To have a better understanding of how well our algorithm works, we performed speaker verification on total of 76 speakers in the

speaker set. The results allow us to know the performance of the algorithm on general speakers.

Table 4.1: Speaker sets used in the experiments

SET	REGION	NUMBER OF MALE	NUMBER OF FEMALE	TOTAL
Speaker set	2	46	30	76
Anti-speaker set	1	20	18	38
Pseudo-imposter set	4	38	30	68
Imposter set	3	53	23	76

Speakers in the anti-speaker set are used in the training process of MRAN for each candidate speaker. For each candidate speaker in the speaker set, we construct an MRAN network for it during the enrollment. The whole training process is supervised. Speech samples obtained from the candidate speaker is used to train the network. Additionally, in order to train the MRAN properly, utterances from other speakers are used to train the network too. The difference is that the output responses for features vectors from the candidate speaker are active, however, the output responses for features vectors from other speakers are inactive. These ‘other speakers’ are included in the anti-speaker set.

As mentioned, the verification threshold for each MRAN network needs to be determined after the network has been trained completely. Feature vectors from pseudo-imposters in the pseudo-imposter set are fed to each network to calculate the corresponding false accept rate. By adjusting the threshold of the MRAN network, different false accept rates are obtained. The verification threshold is the threshold at which the false accept rate is equal to a predefined value.

Finally, data of imposters in the imposter set are used in the verification phase to calculate the false accept rate.

During training, raw speech signals can not be fed into the network directly. Instead, acoustic parameters (feature vectors) extracted from speech signal should be used. Linear predictive (LP) derived cepstral coefficients are one of the best features for speaker recognition [3].

Linear predictive coding (LPC) exploits the fact that important acoustic parameters related to the vocal tract shape may be extracted from the speech signal as infrequently as every 30 milliseconds (called frame size) and still characterize the speech signal accurately. These parameters can be stored. The vocal tract shape parameters are predictor coefficients, and we are going to use those parameters as the feature vectors. Here is how they are determined through the most commonly used LPC algorithm.

Speech signals are sample and quantized, and equations are set up under the assumption that any sample value can be approximated by a linear combination of the past p samples. Mathematically,

$$\hat{s}_n = \sum_{i=1}^p a_i s_{n-i} \quad (4.1)$$

where a_i represents the predictor coefficients, i.e., vocal tract shape parameters. We'd like these p parameters work well for a large number of samples. So the problem becomes: given N samples of speech, estimate a_i ($i = 1, L, p$) that result in the best fit. One of the reasonable definitions for "best fit" is minimum mean squared error. The optimal values can also be regarded as the most probable parameters if it is assumed the distribution of errors is Gaussian and a priori without no restrictions on the values of a_i .

The error at the n^{th} sample is:

$$e_n = s_n - \hat{s}_n \quad (4.2)$$

So the summed squared error over a window of length N is:

$$E = \sum_{n=0}^{N-1} e_n^2 \quad (4.3)$$

By expanding e_n , we have:

$$E = \sum_{n=0}^{N-1} \left(s_n - \sum_{k=1}^p a_k s_{n-k} \right)^2 \quad (4.4)$$

The minimum of summed squared error occurs when the derivative is zero with respect to each of the parameters, a_k . The differentiating equation with respect to a_j and setting equal to zero::

$$\frac{\partial E}{\partial a_j} = 0 = - \sum_{n=0}^{N-1} \left(2 \left(s_n - \sum_{k=1}^p a_k s_{n-k} \right) s_{n-j} \right) \quad (4.5)$$

The value of E is quadratic in each of the a_k , therefore there is a single solution.

Very large values of a_k would lead to poor prediction and hence the solution to

$\frac{\partial E}{\partial a_k} = 0$ must be a minimum.

Rearrange equation (4.5) gives:

$$\sum_{n=0}^{N-1} s_n s_{n-1} = \sum_{k=1}^p a_k \sum_{n=0}^{N-1} s_{n-k} s_{n-j} \quad (4.6)$$

Define the covariance matrix Φ with elements $\Phi_{i,k}$:

$$\Phi_{i,k} = \sum_{n=0}^{N-1} s_{n-i} s_{n-k} \quad (4.7)$$

Now, we can write equation (4.7) as:

$$\Phi_{i,0} = \sum_{k=1}^p \Phi_{i,k} a_k \quad (4.8)$$

The corresponding matrix format is:

$$\Phi_0 = \Phi \mathbf{a} \quad (4.9)$$

So the covariance method solution is obtained by matrix inverse:

$$\mathbf{a} = \Phi^{-1} \Phi_0 \quad (4.10)$$

Once the LP-derived coefficients, i.e., \mathbf{a} in above equations, are obtained. They can be fed into MRAN networks as the input.

In our experiment, the speech signals are first pre-emphasized using a filter with transfer function $H(z) = 1 - 0.95z^{-1}$ (this filter has been widely adopted in speech signal processing). We then compute 12th order (i.e., $p = 12$) LP-derived cepstral coefficients from the speech signals using the algorithm described above. These LP-derived cepstral are used as the feature vectors. Feature vectors extracted from the silent regions do not contain any vocal tract characteristic, thus they are removed.

Now, the feature vectors have been prepared, and we are ready to train the MRAN using them.

4.3 Speaker Enrollment

In the enrollment phase, each candidate speaker in the speaker set is assigned a personalized MRAN network which models the characteristics of his or her voice. Each MRAN network is trained to classify the data derived from two classes – the target speaker class and the anti-speaker set class. For every network, only feature

vectors extracted from the SA and SX sentence sets are used during the training phase. The architecture of the network for a single speaker is illustrated in Figure 4.2.

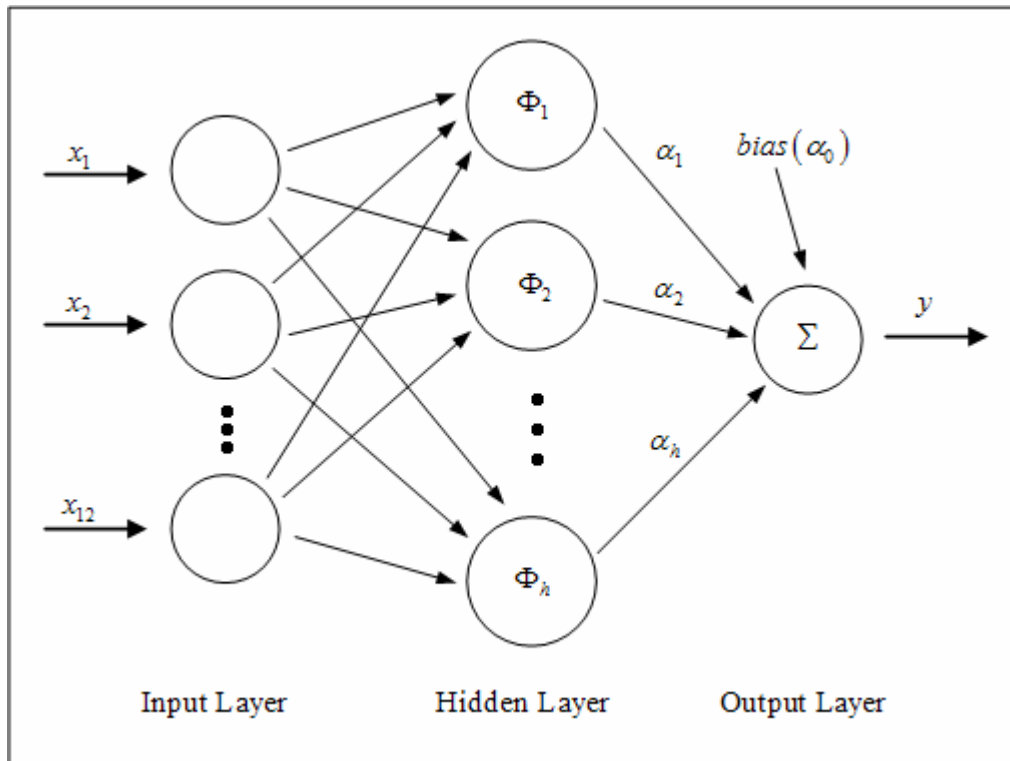


Figure 4.2: Network setup for each speaker in the speaker set

In the input layer, there are 12 input units corresponding to 12th order LP-derived coefficients. There is only one output unit. The number of hidden units in the hidden layer is unknown until the training is completed. During the training, hidden units may be added, pruned or combined. Also, network parameters like weights of hidden units to the output layer are updated during the training process.

The enrollment phase consists of two main steps:

Step 1: For each candidate speaker in the speaker set, use the speaker's data and all anti-speakers' data to train a MRAN network;

Step 2: Determine the decision (acceptance or rejection) threshold using pseudo-imposter set.

The MRAN network setup is shown in Figure 4.2. There are twelve inputs, corresponding to 12th order cepstral coefficients. In the network, there is only one output unit. The training is supervised. For input feature vectors from the target speaker, the output is set to 1; for input feature vectors from the anti-speakers, the output is set to -1 . Note that there are 38 speakers in the anti-speaker set. Since each speaker contributes the same number of sentences (SA and SX sentence set) in the training, the ratio of training vectors between the target speaker class and anti-speaker set class is around 1 to 38. These highly unbalanced inputs create a problem. Due to the large number of training input vectors of anti-speakers, the MRAN network will tend to treat the target speaker data as a mere noise in the whole training set. To prevent this, we repeatedly append the speaker's data to that of each anti-speaker:

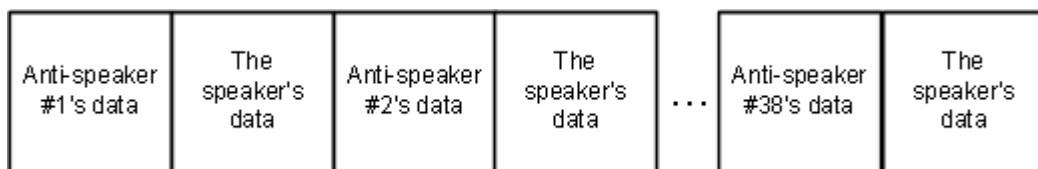


Figure 4.3: The training input data sequence

The training input data sequence is formed as shown in Figure 4.3. The network is then trained using this data sequence.

Before starting to train the network, we need to set several parameters of MRAN properly. These parameters include distance threshold for adding new hidden units, minimum output error for growing, pruning threshold, and many Kalman filter related parameters, etc. The complete list of these parameters and their typical values used in our experiments is shown in table 4.2.

Table 4.2: Typical parameter settings

PARAMETER	VALUE	REMARKS
ε_{\max}	9	Minimum distance to add new hidden unit – initial value
ε_{\min}	0.9	Minimum distance to add new hidden unit – minimum value that can be decayed to
γ	0.99	The decay constant for minimum distance threshold
e_{\min}	1.4	The minimum output error to add new hidden unit
$e_{\min1}$	1	The minimum root mean squared output error over a sliding window
<i>pruningThreshold</i>	0.2	Pruning threshold (normalized)
κ	1.2	Overlap factor for Gaussian functions
<i>RMSWindowSize</i>	30	The size of the sliding window used to calculate the RMS error
<i>pruningWindowSize</i>	30	The size of the sliding window used for pruning criterion determination
R_i	1	(Kalman filter) the variance of measurement noise
P_i	0	(Kalman filter) initial error covariance matrix
P_p	1	(Kalman filter) initial previous error covariance matrix
P_0	1	(Kalman filter) an estimation of the uncertainty in the initial values
<i>randomStepSize</i>	0.25	(Kalman filter) A scalar that determines the allowed random step in the direction of gradient vector

Table 4.2 presents the complete parameter settings for the MRAN network. The values of these parameters are set heuristically largely through trial and error. There are total 14 parameters to be set. However, for this experiment, only four of them are very sensitive that should be set carefully. Other parameters can have typical values as in other MRAN applications. The four sensitive parameters are: ε_{\max} , ε_{\min} , e_{\min} , and $e_{\min1}$.

4.4 Speaker Verification

In this phase, each network trained during enrollment will be verified against the corresponding target speaker and a set of imposters.

In speaker verification, several error rates are used to measure the performance of the verification techniques used [10]. Speaker verification is a detection task, and for a detection system, its performance is usually characterized in terms of two error measures: the miss and false alarm error rates.

False Reject Rate (FRR), also known as miss alarm error rate, measures the probability of not detecting the target speaker when present. FRR is calculated as:

$$E_{FRR} = N_{MISS} / N_{TARGET} \quad (4.11)$$

E_{FRR} is the false reject rate. N_{TARGET} is the number of target speaker trails and N_{MISS} is the number of those where the target speaker was not detected or accepted. False Accept Rate (FAR), also known as false alarm error rate, measuring the probability of falsely detecting the target speaker when not presented. FAR is calculated as:

$$E_{FAR} = N_{FA} / N_{IMPOSTER} \quad (4.12)$$

E_{FAR} is the false accept rate. $N_{IMPOSTER}$ is the number of imposter trials and N_{FA} is the number of those where the target speaker was falsely detected. Another important rate is the Equal Error Rate (EER), which produces a single number performance figure for speaker recognition evaluation. The requirement is that the decision

threshold is adjustable. When the decision threshold is properly adjusted, FAR and FRR can be equal and the value of EER equals to that of FRR or FAR.

During verification, an input feature vector sequence $T = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$ extracted from an utterance spoken by an unknown speaker is fed into the network. The average output can be expressed as:

$$\bar{y} = \frac{1}{t} \sum_{i=1}^t f(\mathbf{x}_i) \quad (4.13)$$

where $f(\mathbf{x}_i)$ is the output of the network with input \mathbf{x}_i .

Verification decisions are made based on the criterion:

$$\begin{cases} \bar{y} > \xi & \Rightarrow \text{accept} \\ \bar{y} \leq \xi & \Rightarrow \text{reject} \end{cases} \quad (4.14)$$

where ξ is the decision threshold determined during the enrollment phase. Decision threshold ξ directly affects the values of FAR and FRR. For example, if ξ is set to 0.99, then the value of FRR is very likely high and the value of FAR will be very low.

In the experiments, we concatenate the feature vectors extracted from the utterances of an unknown speaker to create a test sequence T . The sequence is then divided into many overlapping segments containing 200 consecutive feature vectors (2.8 seconds of speech). The first segment contains the first 200 feature vectors from the sequence. The overlapping window is then moved forward by one vector to get a second segment, which contains feature vectors from number 2 to 201. All the remaining segments can be easily acquired similarly. Verification decisions are then made for all segments using equations (4.13) and (4.14).

To calculate the false reject rate (FRR), feature vectors extracted from SI sentence set spoken by the target speaker are used to form the test sequence. By using equation (4.11), FRR can be calculated by dividing the number of rejected cases over the total number of trails. Similarly, feature vectors derived from the sentences spoken by the speakers in the imposter set are used to calculate false accept rate (FAR). According to equation (4.12), FAR equals to number of accepted cases over total number of imposter trails.

4.5 Decision Threshold

In the verification phase, the decision threshold is used to make verification decisions. By comparing the average network output \bar{y} with the decision threshold ξ , a verification decision can be made. If \bar{y} is larger than the threshold ξ , the unknown speaker is accepted, otherwise, the unknown speaker is rejected.

The decision threshold is determined in the enrollment phase. After the network has been trained using data from the target speaker and anti-speakers, the decision threshold is determined by verifying a series of pseudo-imposters. When utterances of a pseudo-imposter are input to the network, the average network output can be computed. If the average network output for a pseudo-imposter is less than the decision threshold, then the network successfully verifies the pseudo-imposter, otherwise, the verification fails. The failure ratio is the false alarm rate (FAR). By adjusting the decision threshold, we can achieve a reasonable FAR (e.g., 2%). Thus, we can use the decision threshold determined to perform speaker verification later.

The following steps are used to determine the proper decision threshold for each MRAN network:

- 1) First set decision threshold to a very small value (e.g., -1);

- 2) Speakers from the pseudo-imposter set are verified against the network, and the FAR rate can be calculated.
- 3) If FAR is larger than the predefined value (e.g., 2%), then increase the decision threshold repeat step (2) until FAR is equal to the predefined value.

Figure 4.4 shows the FAR and FRR versus the decision threshold.

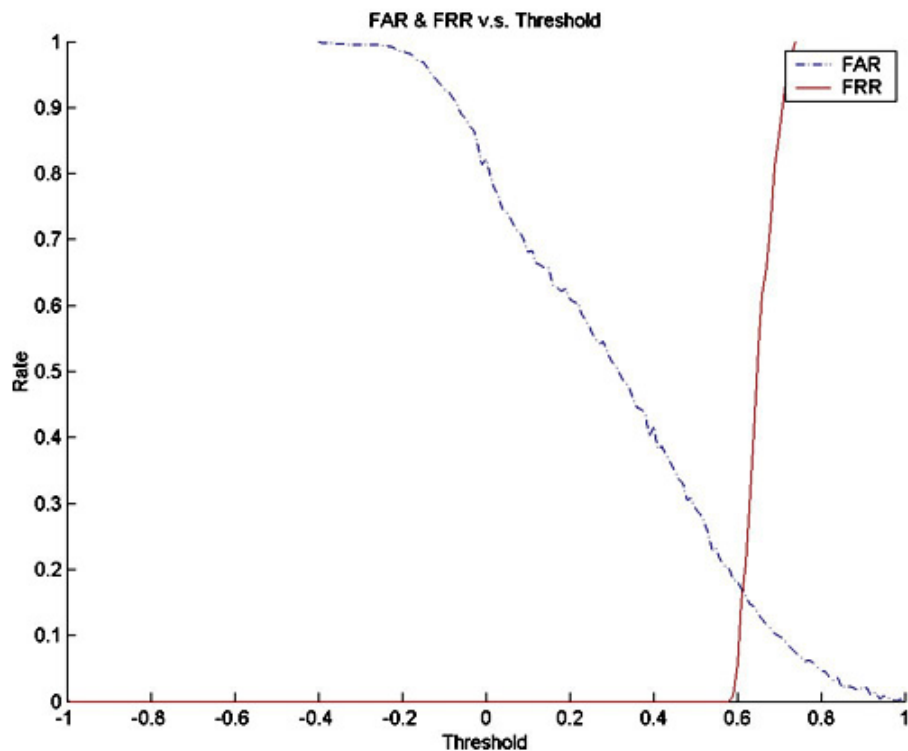


Figure 4.4: FAR and FRR versus the decision threshold of a MRAN network

Determining a decision threshold using the above procedure takes considerable time. So we have come out a fast search technique which determines the decision thresholds more accurately with much less time. The following steps are involved in this technique:

- 1) Set two variables *min* and *max*. Initially: $min = -1$, $max = 1$ (obviously, the decision threshold should be in the range $[-1, 1]$);

- 2) Calculate the FAR when decision threshold is set to $(max + min)/2$:
 - If the FAR is larger than the predefined value, set $min = (max + min)/2$;
 - If the FAR is less than the predefined value, set $max = (max + min)/2$;
 - If the FAR equals to the predefined value, then decision threshold is found: $(max + min)/2$;
- 3) Repeat step (2) until decision threshold is found.

Compared with the first procedure, this binary search technique works faster and more accurate. The total time consumed for threshold determination is reduced significantly.

4.6 Experimental Results

In [29], Mak and Kung applied radial basis function (RBF) networks and elliptical basis function (EBF) networks to the speaker verification problem. A RBF network is trained in a traditional way. First k-means algorithm is used to find hidden centers for the speaker and anti-speakers. The RBF function widths corresponding to hidden centers in the network are then computed using the K-nearest neighbor's algorithm. Finally, Singular Value Decomposition (SVD) is used to find the weights of the hidden units to the output layer. They abbreviated the EBF network with covariance matrices approximated by the sample covariance¹ as "EC". To improve the accuracy of the EBF network, they use a second EBF network with covariance matrices estimated using the Expectation-Maximization (EM) algorithm, where is

¹ This method derives covariance matrix from the input feature vectors (referred to as "sample covariance") and use it as the covariance matrix for the elliptical basis function.

referred as “EEF”. Another EBF with dialog covariance estimated using the EM algorithm (“EED”) is also proposed. Procedures of calculating hidden centers and weights of all the EBF networks are the same as those of the RBF network. Mak and Kung applied the four kinds of networks (the RBF network, EC, EEF, and EED) described above on speaker verification.

In our experiments, we use the MRAN networks to model speakers. We have done total of 76 experiments covering all the speakers in the speaker set using the procedures described in previous sections. The experimental results are then compared with the results by Mak and Kung [29]. Note that we use the same setup and speaker data as those in [29].

In speaker verification, it is very hard to compare different systems using multi-error rates. It is desirable to compare different algorithms in terms of a single error rate that represents their effectiveness to discriminate among speakers. The geometric mean error (GME) is a statistic offering the desired ability of comparing systems, which is defined as:

$$E_{GME} = \sqrt{E_{FAR} * E_{FRR}} \quad (4.15)$$

where E_{FAR} is the false accept rate and E_{FRR} is the false reject error.

One of the drawbacks of GME is that it tends to increase as the FRR decreases [10]. This effect becomes more significant when the FRR goes down to around 2%. Nonetheless, we will use GME as one of the measures for comparison.

Table 4.3: Comparison of Results

	RBF	MRAN	EC	EED	EEF
Hidden Units	61	57	10	35	10
Complexity	917	804	922	912	922
EER	7.46%	0.95%	0.44%	0.47%	0.37%
FAR	20.72%	3.18%	0.46%	1.78%	3.52%
FRR	53.93%	5.30%	14.0%	15.1%	6.74%
GME	33.43%	4.10%	2.62%	5.18%	4.87%

Table 4.3 compares our speaker verification results with those in [29]. All the values in table 4.3 are average values over 76 networks corresponding to 76 speakers in the speaker set, regardless of network type. In the table, the complexity of the network is expressed in number of parameters used in it. The abbreviations used in the table are explained as follows.

- **RBF** – the RBF network with K-means algorithm for hidden center determination in [29];
- **MRAN** – the MRAN network used in our experiments;
- **EC** – the EBF network with K-means algorithm for hidden center determination and sample covariance for covariance matrices approximation in [29];
- **EED** – the EBF network with K-means algorithm for hidden center determination and expectation maximization algorithm for diagonal covariance estimation in [29];
- **EEF** – the EBF network with K-means algorithm for hidden center determination and expectation maximization algorithm for full covariance estimation in [29].

From Table 4.3, it is easy to notice that the RBF in [29] performs significantly poorer than any of the EBFs used in [29]. However, the results show that our MRAN network has outperformed the RBF and EBFs with diagonal and full covariance matrices (EED and EEF) in terms of geometric mean error (GME) and complexity, i.e., number of network parameters. The average equal error rate (EER) of the MRAN is much less than that of RBF used in [29], however, it is slightly larger than those of EBFs. But MRAN's complexity in terms of number of network parameters is much less as compared to any of the networks regardless RBF or EBF used in [29]. The GEM error of MRAN is slightly higher than that of EC, but EC is more complex than MRAN.

Table 4.3 clearly shows that MRAN produces comparable error rates to other methods with much less computational complexity. In Table 4.3, the complexity of a network is expressed in the number of parameters used in it. Next section provides a detailed computational complexity analysis and comparison of MRAN and networks used in [29].

4.7 Computational Complexity Analysis

This section provides a detailed computational complexity analysis of all the networks mentioned in previous section. Before going into details, Table 4.4 shows a set of notations that will be used throughout this section:

Table 4.4: Notations used in complexity computation

<i>NOTATION</i>	<i>MEANING</i>
N_{sp}	Total number of training feature vectors from the target speaker
N_{anti}	Total number of training feature vectors from the speakers in the anti-speaker set
D_{input}	Dimension of the input vectors (12 in our case)
D_{output}	Dimension of the output vectors (1 in our case)
H_{sp}	Total number of hidden neurons allocated for the speaker
H_{anti}	Total number of hidden neurons allocated for anti-speakers
$H = H_{sp} + H_{anti}$	Total number of hidden neurons
S	(average) number of input vectors that belongs to a hidden unit in the K-means algorithm

For all the networks in [29], Mak and Kung use K-means to find a set of hidden units representing the speaker and a set of hidden units representing anti-speaker separately. The total number of hidden neurons allocated for the speaker is represented by H_{sp} , and H_{anti} represents the total number of hidden neurons allocated for anti-speakers. While for MRAN, we take a different approach, and we let the network to allocate the hidden neurons automatically from the input. Thus one can not easily tell which hidden units in MRAN model the speaker and which model the anti-speakers. The total number of hidden neurons in MRAN is denoted as H .

4.7.1 RBF

The training of a RBF for a speaker in [29] involves the following steps:

1. Applying the K-means algorithm to the cepstral vectors of the speaker to find the hidden unit for the speaker;
2. Applying the K-means algorithm to the cepstral vectors of the anti-speakers to find the hidden unit for anti-speakers;
3. Applying the K-nearest neighbors algorithm (with K=2) to the hidden centers to find the function widths corresponding to the hidden centers;
4. Computing the smoothing parameter controlling the spread of the basis function γ_j according to $\gamma_j = \frac{3}{5} \sum_{k=1}^5 \|\mu_k - \mu_j\|$ (μ_k denotes the k^{th} nearest neighbor of μ_j);
5. Applying Singular Value Decomposition (SVD) to find the output weights.

According to [15], the complexity for K-means algorithm is $O(kmni)$, where k is the number of clusters, m is the number of attributes for each instance, n is the number of instances (i.e., training patterns), and i is the number of iterations. So the complexity for step 1 will be:

$$O(kmni) = O(H_{sp} N_{sp} D_{input} * i) \quad (4.16)$$

i , the number of iterations performed, can be approximated as $i \approx 0.2H_{sp}$ [14]. So the complexity of the K-means algorithm to find the hidden centers for the speaker is:

$$O_{step1} = O(H_{sp}^2 N_{sp} D_{input}) \quad (4.17)$$

Similarly, the computational complexity of step 2 can be computed:

$$O_{step2} = O(H_{anti}^2 N_{anti} D_{input}) \quad (4.18)$$

The computational complexity of step 3, applying K-nearest algorithm to find function widths, is:

$$O_{step3} = O\left(H_{sp} N_{sp} D_{input} + H_{anti} N_{anti} D_{input}\right) \quad (4.19)$$

Step 4 requires:

$$O_{step4} = O\left(HD_{input}\right) \quad (4.20)$$

The SVD algorithm used to calculate the weights with least squared error in step 5 is very computation intensive, which requires [11]:

$$O_{step5} = O\left(N^2 D_{output} + ND_{output}^2 + D_{output}^3\right) \quad (4.21)$$

By summing the complexity amounts for steps 1-5, we have:

$$\begin{aligned} O_{RBF} &= O_{step1} + O_{step2} + O_{step3} + O_{step4} + O_{step5} \\ &= O\left(H_{sp}^2 N_{sp} D_{input}\right) + O\left(H_{anti}^2 N_{anti} D_{input}\right) + \\ &O\left(H_{sp} N_{sp} D_{input} + H_{anti} N_{anti} D_{input}\right) + O\left(HD_{input}\right) + \\ &O\left(N^2 D_{output} + ND_{output}^2 + D_{output}^3\right) \end{aligned} \quad (4.23)$$

(4.23) can be simplified into:

$$O_{RBF} = O\left(H^2 ND_{input} + N^2 D_{output} + ND_{output}^2 + D_{output}^3\right) \quad (4.24)$$

4.7.2 EBF with Sample Covariance

Training of the EBF network with sample covariance in [29] involves the following steps:

1. same as step (1) for RBF;

2. same as step (2) for RBF;
3. Applying $\Sigma_j \approx \hat{\Sigma}_j = \frac{1}{N_j} \sum_{x \in X_j} (x - \mu_j)(x - \mu_j)^T$ to obtain the function widths;
4. same as step (4) for RBF;
5. same as step (5) for RBF;

The complexity for step 3 is:

$$O_{step3} = O(HD_{input}) \quad (4.25)$$

So the total complexity will be:

$$O_{EC} = O(H^2ND_{input} + N^2D_{output} + ND_{output}^2 + D_{output}^3) \quad (4.26)$$

4.7.3 EBFs with EM algorithm

Training of the EBF network with full covariance in [29] involves the following steps:

1. same as step (1) for RBF;
2. same as step (2) for RBF;
3. same as step(3) for RBF;
4. Applying the EM algorithm to update network parameters;
5. same as step (4) for RBF;
6. same as step (5) for RBF;

The computational complexity for the EM algorithm can be calculated as follows:

For each iteration (i.e., for each training pattern), the following formulas are used to update network parameters [29].

$$p(\mathbf{x}^r | j) = \frac{1}{(2\pi)^{\frac{I}{2}} \prod_{i=1}^I \sigma_{ji}^{old}} \exp \left\{ -\frac{1}{2} \sum_{i=1}^I \frac{(x_i - \mu_{ji}^{old})^2}{(\sigma_{ji}^{old})^2} \right\} \quad (4.27)$$

$$P^{old}(j | \mathbf{x}^r) = \frac{p(\mathbf{x}^r | j) P^{old}(j)}{\sum_k p(\mathbf{x}^r | k) P^{old}(k)} \quad (4.28)$$

$$\mu_j^{new} = \frac{\sum_{\mathbf{x}^r \in X} P^{old}(j | \mathbf{x}^r) \mathbf{x}^r}{\sum_{\mathbf{x}^r \in X} P^{old}(j | \mathbf{x}^r)} \quad (4.29)$$

$$\Sigma_j^{new} = \frac{\sum_{\mathbf{x}^r \in X} P^{old}(j | \mathbf{x}^r) (\mathbf{x}^r - \mu_j^{new})(\mathbf{x}^r - \mu_j^{new})^T}{\sum_{\mathbf{x}^r \in X} P^{old}(j | \mathbf{x}^r)} \quad (4.30)$$

$$P^{new}(j) = \frac{\sum_{\mathbf{x}^r \in X} P^{old}(j | \mathbf{x}^r)}{\sum_{r=1}^J N_r} \quad (4.31)$$

where I is the total number of hidden neurons. The complexity amount required for each formula about per each hidden unit is:

$$O(p(\mathbf{x}^r | j)) = O(D_{input} + D_{input} + D_{input}) = O(D_{input}) \quad (4.32)$$

$$O(P^{old}(j | \mathbf{x}^r)) = O(H) \quad (4.33)$$

$$O(\mu_j^{new}) = O(S * D_{input} + S) = O(S * D_{input}) \quad (4.34)$$

$$O(\Sigma_j^{new}) = O(S * (1 + D_{input} + D_{input}^2) + S) = O(S * D_{input}^2) \quad (4.35)$$

$$O(P^{new}(j)) = O(S) \quad (4.36)$$

So the total complexity of step (4) for the full covariance EBF network will be:

$$\begin{aligned}
O_{step4(Full)} &= NH \left[O(D_{input}) + O(H) + O(S * D) + O(SD_{input}^2) + O(S) \right] \\
&= O(NHSD_{input}^2)
\end{aligned} \tag{4.37}$$

For the diagonal covariance EBF network, the total complexity of step (4) is:

$$\begin{aligned}
O_{step4(Diagonal)} &= NH \left[O(D_{input}) + O(H) + O(S * D) + O(SD_{input}) + O(S) \right] \\
&= O(NHSD_{input})
\end{aligned} \tag{4.38}$$

The total computational complexity needed to train a full covariance EBF is:

$$\begin{aligned}
O_{EBF(Full)} &= O(H^2ND_{input}) + O(HN) + O(NSD_{input}^2) + \\
&O(N^2D_{output} + ND_{output}^2 + D_{output}^3)
\end{aligned} \tag{4.39}$$

and the total training complexity for a diagonal covariance EBF is:

$$\begin{aligned}
O_{EBF(Diagonal)} &= O(H^2ND_{input}) + O(HN) + O(NSD_{input}) + \\
&O(N^2D_{output} + ND_{output}^2 + D_{output}^3)
\end{aligned} \tag{4.40}$$

4.7.4 MRAN

For MRAN network, the algorithm goes as follows:

For each observation

$$1) \text{ Calculate the network output using } f(x_n) = \alpha_0 + \sum_{k=1}^h \alpha_k \Phi_k(x)$$

the complexity is $O(H * D_{input})$

2) Calculate e_{msn} , the corresponding complexity is: $O(1)$

3) Update network parameters with EKF:

$$O(EKF) = O\left(\left(H * (D_{input} + 2)\right)^2\right) = O\left(H^2 D_{input}^2\right)$$

4) Prune:

$$o_k = \alpha_k \Phi_k(x)$$

$$O(o_k) = H * D_{input}$$

Find the largest absolute hidden unit output value. $O(H)$

Compute normalized output values. $O(H)$

Pruning: $O(H)$

End for each observation

The complexity of the MRAN is:

$$\begin{aligned} O_{MRAN} &= O(NHD_{input}) + O(N) + O(NH^2 D_{input}^2) + O(NHD_{input}) + O(H) \\ &= O(NH^2 D_{input}^2) \end{aligned} \quad (4.41)$$

4.7.5 Comparison

Now, the training computation complexity for each network is computed. Before comparing them, we need to make some reasonable assumptions. Normally, the values of dimensions of input vectors and output vectors are orders of magnitude smaller than the value of total number of training sets. For example, in all the experiments described in Chapter 4, the input dimension is 12, and the output dimension is 1. However, the total number of training sets is as large as 30,400. For a particular set of problems, the dimensions are known prior to the training. For the sake of simplicity, we treat the dimensions of input vectors and output vectors as

constants and thus dimension parameters are omitted. The computational complexity of MRAN and those neural networks in [29] is compared in Table 4.5, where H stands for the number of hidden units and N stands for the total number of training data.

Table 4.5: Comparison of computational complexity

NETWORK	COMPLEXITY
RBF	$O(H^2N + N^2)$
EC	$O(H^2N + N^2)$
EED	$O(H^2N + N^2)$
EEF	$O(H^2N + N^2)$
MRAN	$O(H^2N)$

Table 4.5 shows that each of the networks in [29] requires $O(H^2N + N^2)$ operations mainly due to the computation intensive SVD procedure. However, our MRAN network needs only $O(H^2N)$ operations. For MRAN, the training time is approximately proportional to the size of training set; however, for all the networks in [29], the training time is approximately proportional to the square of the training set size. The comparison shows that MRAN has much less computational complexity than those of other RBF/EBF networks during the training phase.

Chapter 5

Extension of MRAN to MRAN-EBF

The previous chapter shows that MRAN produces comparable error rates to other well-known methods but with much less computational complexity. MRAN is a RBF Gaussian network. In [29], Mak and Kung showed that their small EBF networks with basis function parameters estimated by the EM algorithm perform better than large RBF networks in term of equal error rate. In the hope of finding more efficient speaker verification algorithms, we extended MRAN from RBF Gaussian to elliptical basis functions. The resulting network is called MRAN-EBF. MRAN-EBF is then applied to speaker verification and compared with other algorithms.

5.1 MRAN-EBF

The architecture of MRAN-EBF (shown in Figure 5.1) is very similar to that of MRAN. The main difference is that the MRAN-EBF uses elliptical basis functions (EBF) as the basis function for hidden units while MRAN uses radial basis functions. In Figure 5.1, $\Phi_1, \Phi_2, \dots, \Phi_h$ represent the elliptical basis functions for the

hidden units. The output $y = f(\mathbf{x})$ of the network is a linear combination of all the outputs of the elliptical basis functions:

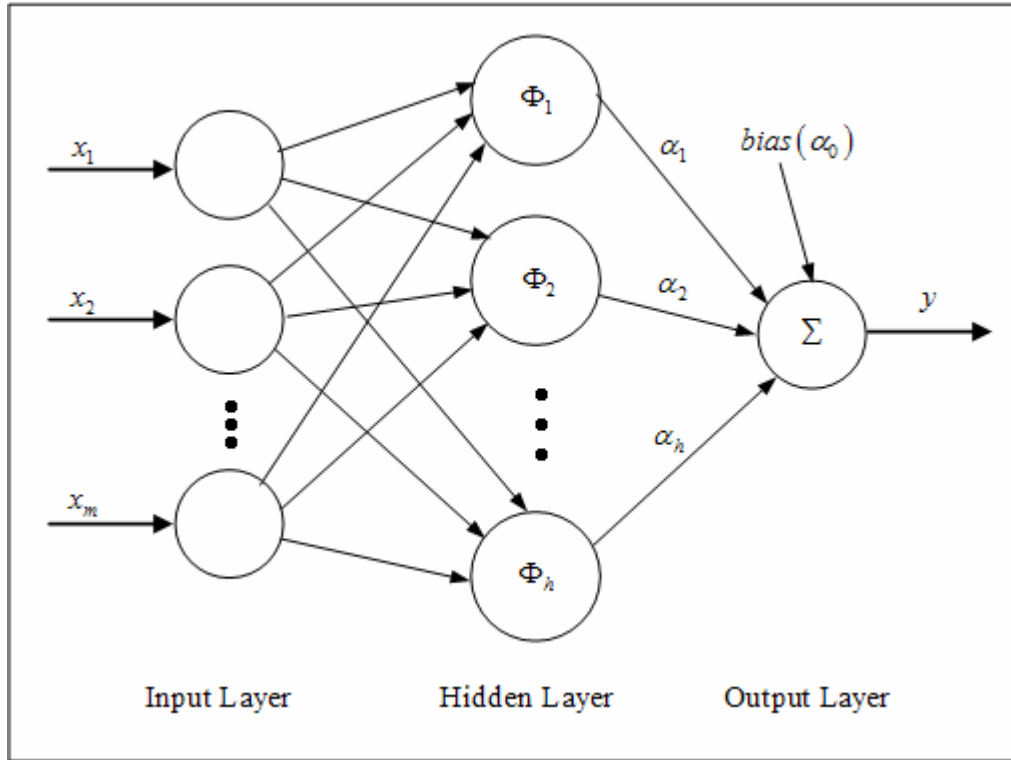


Figure 5.1: Architecture of MRAN-EBF

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^h \alpha_k \Phi_k(\mathbf{x}) \quad (5.1)$$

where coefficient α_k is the connecting weight of the k^{th} hidden neuron to the output unit, α_0 is the bias term and Φ_k is the elliptical basis function for the k^{th} hidden neuron.

The difference between MRAN and MRAN-EBF is that MRAN-EBF uses full covariance Gaussian function while MRAN uses a simplified version of Gaussian function.

5.1.1 Gaussian Functions

The Gaussian (normal) distribution density function for the case of a single variable can be written as:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (5.2)$$

where σ^2 and μ are the variance and mean, which satisfy:

$$\mu = E[x] = \int_{-\infty}^{\infty} xp(x) dx \quad (5.3)$$

$$\sigma^2 = E[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x) dx \quad (5.4)$$

where $E[g]$ represents the expectation.

Extending formula (5.2) to multiple dimensions, we can get the general multivariate Gaussian probability density:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \quad (5.5)$$

where d is the number of dimensions; $\boldsymbol{\mu}$ is the mean, a d -dimensional vector; Σ is the covariance matrix, a $d \times d$ matrix; and $|\Sigma|$ is the determinate of Σ . The mean and covariance matrix satisfy:

$$\boldsymbol{\mu} = E[\mathbf{x}] \quad (5.6)$$

$$\Sigma = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T] \quad (5.7)$$

Formula (5.7) suggests that Σ is a symmetric matrix, which has $d(d+1)/2$ independent components. The mean vector $\boldsymbol{\mu}$ has d independent elements. Thus, to specify the Gaussian density function (5.5), we need to determine the total of $d(d+3)/2$ parameters.

The Gaussian distribution function can be simplified by using diagonal covariance matrix, i.e.,

$$(\Sigma)_{ij} = \delta_{ij} \sigma_j^2 \quad (5.8)$$

In this case, the total number of independent parameters in the distribution is reduced to $2d$.

Further simplification can be made by setting the all the diagonal elements in the diagonal covariance matrix to be the same value, thus the total number of parameters is further reduce to $d+1$:

$$\sigma_j = \sigma \quad \text{for all } j \quad (5.9)$$

The Gaussian radial basis function used in MRAN uses (5.9) simplification. For the MRAN-EBF, the original un-simplified Gaussian distribution density function is used.

5.1.2 From MRAN to MRAN-EBF

Similar to MRAN, the steps of MRAN-EBF training involves:

1. For each input \mathbf{x} : calculate the network output y ;
2. Create a new hidden unit if all of the three criteria are met;

3. Adjust network parameters (centers, covariances, weights of all existing hidden neurons) using the Extended Kalman Filter if no hidden neuron is added;
4. Prune hidden neurons whose contribution to the network output for a certain number of consecutive inputs is below the pruning threshold;
5. Repeat steps 1-4 till the last input data.

For MRAN-EBF, the network output is:

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^h \alpha_k \Phi_k(\mathbf{x}) \quad (5.10)$$

where Φ_k is the elliptical basis function, given by:

$$\Phi_k(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T C_k (\mathbf{x} - \boldsymbol{\mu}_k)\right) \quad (5.11)$$

where $C_k = \Sigma_k^{-1}$, and C_k is a symmetric matrix.

When an input data does not meet all of the adding new hidden unit criteria, the network parameters will be updated using Extended Kalman Filter (EKF). All of the network parameters are put into a matrix:

$$W = \left[\alpha_0^T, \alpha_1^T, \mu_1^T, \{C_1\}, L, \alpha_h^T, \mu_h^T, \{C_h\} \right]^T \quad (5.12)$$

where α_0 is the bias term; α_k ($k = 1, L, h$) is the weight of k^{th} basis function to the output layer; μ_k ($k = 1, L, h$) is the center of k^{th} basis function; and $\{C_k\}$ ($k = 1, L, h$) represents the elements in the covariance matrix:

$$\{C_k\} = \underset{d(d+1)/2 \text{ elements}}{\begin{matrix} c_{11}^k, c_{12}^k, \dots, c_{1L}^k, \\ c_{21}^k, c_{22}^k, \dots, c_{2L}^k, \\ \vdots \\ c_{L1}^k, c_{L2}^k, \dots, c_{LL}^k \end{matrix}} \quad (5.13)$$

Expanding (5.12), we have:

$$W = \begin{bmatrix} \alpha_{10}, \alpha_{20}, \mathbf{L}, \alpha_{p0}, \\ \alpha_{11}, \alpha_{12}, \mathbf{L}, \alpha_{1p}, \quad \mu_{11}, \mu_{12}, \mathbf{L}, \mu_{1d}, \quad c_{11}^1, c_{12}^1, \mathbf{L}, c_{d(d+1)/2}^1, \\ \mathbf{L} \\ \alpha_{h1}, \alpha_{h2}, \mathbf{L}, \alpha_{hp}, \quad \mu_{h1}, \mu_{h2}, \mathbf{L}, \mu_{hd}, \quad c_{11}^h, c_{12}^h, \mathbf{L}, c_{d(d+1)/2}^h \end{bmatrix}^T \quad (5.14)$$

where d , p , h represent the number of input dimension, number of output dimension, and total number of hidden neurons in the network, respectively. W is a $z \times p$ matrix, where:

$$z = p + h[p + d + d(d+1)/2] = p(h+1) + hd(d+3)/2 \quad (5.15)$$

So the gradient of the function $f(\mathbf{x})$ with respect to the parameter vector W can be written as:

$$B_{(z \times p)} = \nabla_W f(\mathbf{x}) \quad (5.16)$$

To calculate $B_{(z \times p)}$, we can compute each component and then concatenate them to get the result:

$$\nabla_{\alpha_0^T} f(\mathbf{x}) = I_{(p \times p)} \quad (5.17)$$

$$\nabla_{\alpha_h^T} f(\mathbf{x}) = \Phi_h(\mathbf{x}) I_{(p \times p)} \quad (5.18)$$

$$\nabla_{\mu_h^T} f(\mathbf{x}) = \Phi_h(\mathbf{x}) \alpha_h [\mathbf{x} - \boldsymbol{\mu}]^T C \quad (5.19)$$

$$\nabla_{c_{jk}^h} f(\mathbf{x}) = \begin{cases} \Phi_h \alpha_h \left(\frac{1}{2} \right) (x_j - \mu_j)^2 & \text{if } j = k \\ \Phi_h \alpha_h (-1) (x_j - \mu_j) (x_k - \mu_k) & \text{if } j \neq k \end{cases} \quad (5.20)$$

Now, $B_{(z \times p)}$ has been computed and we can use it in EKF to update network parameters,

All other procedures of MRAN-EBF are the same as those of MRAN.

5.2 Application of MRAN-EBF to Speaker Verification

In this chapter, we set up experiments to compare the performance of MRAN-EBF with that of MRAN for speaker verification problems.

The basic experiment setup is similar to that of last chapter. For each candidate speaker, we construct a MRAN-EBF network and train it. After the network has been properly trained, we then determine its verification threshold. Finally, we perform some speaker verification tasks and record its performance in terms of false accept rate, false reject rate and equal error rate. The average error rates for MRAN-EBF can then be calculated. Now, we repeat the same procedures using of MRAN instead of MRAN-EBF. We then compare the performance of MRAN-EBF with that of MRAN.

Table 5.1: Speaker sets used in the experiments

SET	REGION	NUMBER OF MALE	NUMBER OF FEMALE	TOTAL
Speaker set	2	10	10	20
Anti-speaker	1	10	10	20
Pseudo-imposter set	4	15	15	30
Imposter set	3	20	10	30

The speaker sets used in this experiment are shown in Table 5.1.

Totally 100 speakers from the first four regions of the TIMIT corpus is used. These speakers are divided into four sets as shown in Table 5.1. The speaker set contains all the candidate speakers. Speakers in the anti-speaker set are used in the training process of each candidate speaker. For each candidate speaker in the speaker set, we use a network for it during the enrollment. The whole training process is supervised. Speech inputs from the candidate speaker result active responses and those from the anti-speakers result inactive responses.

After a network has been properly trained, data from pseudo-imposters is used to determine the verification threshold for the network. Feature vectors from pseudo-imposters in the pseudo-imposter set are input to the network to obtain the corresponding false accept rate. By adjusting the threshold of the network, different false accept rates can be achieved. The verification threshold is the threshold at which the false accept rate is equal to a predefined value. Finally, in the last stage, data of imposters in the imposter set are used to calculate the false accept rate.

Again, linear predictive (LP) derived cepstral coefficients are used as feature vectors. 12th order LP-derived cepstral coefficients are extracted from the utterances. Feature vectors extracted from the silent regions are then removed.

In the enrollment phase, data from the candidate speaker and anti-speakers are used to train the MRAN-EBF or MRAN network. Various parameters settings are shown in Table 5.2.

After the training of the network has been completed, the verification threshold is determined by using data from pseudo-imposters. By adjusting the threshold, different false accept rates can be achieved. The verification decision threshold for the network is set to the threshold value at which the false accept rate is equal to 2%.

Table 5.2: Typical parameters settings for MRAN-EBF and MRAN

PARAMETER	MRAN	MRAN-EBF
ϵ_{\max}	9	20
ϵ_{\min}	0.9	1.5
γ	0.99	0.99
e_{\min}	1.4	0.7
$e_{\min 1}$	1	1
<i>pruningThreshold</i>	0.2	0.25
κ	1.2	-
<i>RMSWindowSize</i>	30	30
<i>pruningWindowSize</i>	30	30
R_i	1	1
P_i	0	0
P_p	1	1
P_0	1	1
<i>randomStepSize</i>	0.25	0.25

Finally, we use the data from the speaker and the imposters to perform a series of speaker verification and compute the false reject rate and false accept rate of the network.

During the verification phase, for an input feature vector sequence $T = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$ extracted from an utterance spoken by an unknown speaker is input to the network. The average network output is:

$$\bar{y} = \frac{1}{t} \sum_{i=1}^t f(\mathbf{x}_i) \quad (5.20)$$

where $f(\mathbf{x}_i)$ represent the network output for input \mathbf{x}_i . The verification decisions are made based on the criterion:

$$\begin{cases} \bar{y} > \xi & \Rightarrow \text{accept} \\ \bar{y} \leq \xi & \Rightarrow \text{reject} \end{cases} \quad (5.21)$$

where ξ is the verification threshold for the network.

The false reject rate (FRR) can be calculated by using feature vectors extracted from the SI sentence set spoken by the candidate speaker. The false reject rate equals to the number of rejected cases divided by the total number of trials. Similarly, feature vectors derived from the sentences spoken by the speakers in the imposter set are used to calculate the false accept rate. The false accept rate (FAR) equals to number of accepted cases over total number of imposter trials.

Now, the FRR and FAR rates are obtained. We can use them to compare the performance of MRAN-EBF and MRAN.

The comparison of MRAN-EBF and MRAN is cover in next section very soon. One of the MRAN-EBF's drawbacks is that the complexity of the MRAN-EBF network is much higher than that of a MRAN in terms of network parameters for the same number of hidden neurons. More network parameters means longer training and verification time. In view of this, we also tried to use MRAN-EBF with diagonal covariance matrix to perform speaker verification. The results of MRAN-EBF with diagonal covariance matrix are also compared with those of MRAN-EBF and MRAN in next section.

5.3 Experimental Results

This section presents and compares the experimental results from the three networks: MRAN, MRAN-EBF, and MRAN-EBF with diagonal covariance matrix.

Table 5.3 summarizes the false acceptance rates (FAR's), false rejection rates (FRR's) and numbers of hidden neurons for MRAN, MRAN-EBF and MRAN-EBF with diagonal covariance matrix. All the values in Table 5.3 are based on the average of 20 speakers. The complexity of network is expressed in number of network parameters used in it.

Table 5.3: FAR's, FRR's and numbers of hidden neurons for MRAN, MRAN-EBF, and MRAN-EBF with diagonal covariance.

	MRAN	MRAN-EBF	MRAN-EBF WITH DIAGONAL COVARIANCE MATRIX
FAR	3.10%	37.57%	3.84%
FRR	4.81%	51.40%	5.01%
Number of hidden neurons	58	55	61
Complexity	812	5005	1375

Table 5.3 clearly shows that MRAN-EBF performs worse (in terms of FAR and FRR) than MRAN and with a much higher of complexity. MRAN-EBF with diagonal covariance matrix produces comparable experimental results with MRAN. MRAN-EBF with diagonal covariance matrix performs slightly worse than MRAN in terms of FAR and FRR and MRAN-EBF with diagonal covariance matrix has a higher of complexity.

Some of the possible reasons for the poor performance of MRAN-BEF are:

i) First, the training data set is too small compared with the huge number of independent network parameters for a MRAN-EBF. The number of training feature vectors from a candidate speaker is around 400 and the number of training feature vectors from the twenty anti-speakers is around 8000. This amount of training input may be sufficient for MRAN; however, it would be inadequate for a MRAN-EBF network with 5000 plus network parameters.

ii) EKF produces highly unstable estimates when the assumptions of local linearity is violated leading to significant errors [40]. This effect may not be significant for MRAN. However, for MRAN-EBF with a large number of parameters, it could become so significant that it produces very large prediction error.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

Minimal resource allocation network (MRAN) is a sequential learning algorithm for radial basis function networks, in which hidden neurons are added or pruned as training progress. In this research project, we have successfully applied MRAN to perform speaker verification tasks. Experiments have been done to evaluate the performance of MRAN. Experimental results show that MRAN produces comparable error rates to other methods but with much less computational complexity.

The experiment involves four phases. First, utterances spoken by the speaker are translated into linear predictive coefficients. These coefficients are used as the feature vectors. The second phase is the training phase. Feature vectors extracted from the speaker and anti-speakers are fed into the network to train the network. This training process is supervised. After training, a third phase is used to determine the verification threshold. The verification threshold is used to make the decision of

accepting or rejecting the identity claim based on the output or the average output value of a certain number of outputs. The final phase is used to evaluate the performance of the network. There are several rates that can be used to measure the performance of a speaker model, namely, false accept rate, false reject rate, and equal error rate, and geometric mean error.

Table 6.1 summarizes the experimental results.

Table 6.1: Experimental results of MRAN

	RBF	MRAN	EC	EED	EEF
Complexity	917	804	922	912	922
EER	7.46%	0.95%	0.44%	0.47%	0.37%
FAR	20.72%	3.18%	0.46%	1.78%	3.52%
FRR	53.93%	5.30%	14.0%	15.1%	6.74%
GME	33.43%	4.10%	2.62%	5.18%	4.87%

RBF, EC, EED, EEF refers to the RBF network, the EBF network with sample covariance, the EBF network with diagonal covariance matrix, and the EBF network with full covariance matrix used in [29], respectively. The complexity in the table is expressed in the number of network parameters. All the values in Table 6.1 are average values over 76 networks corresponding to 76 speakers in the speaker set, regardless of network type.

The results show that MRAN outperforms the RBF and EBFs with diagonal and full covariance matrices (EED and EEF) in terms of geometric mean error (GME) and complexity, i.e., number of network parameters. The average equal error rate (EER) of MRAN is much less than that of RBF, however, it is slightly larger than those of EBFs. But MRAN's complexity in terms of number of network parameters is the least among all the networks considered.

We have also analyzed the computational complexity for each network. Table 6.2 briefs the computational complexity of each network considered. Due to the computation intensive Singular Value Decomposition (SVD) procedure, the RBF, EC, EED, and EEF networks used in [29] requires as many as $O(H^2N + N^2)$ operations. However, MRAN requires significantly less operations, $O(H^2N)$. The comparison shows that MRAN have much less computational complexity than those of other RBF/EBF networks during the training phase.

Table 6.2: Computational complexity of MRAN and other networks

NETWORK	COMPUTATIONAL COMPLEXITY
RBF	$O(H^2N + N^2)$
EC	$O(H^2N + N^2)$
EED	$O(H^2N + N^2)$
EEF	$O(H^2N + N^2)$
MRAN	$O(H^2N)$

In hope of finding more efficient speaker verification algorithms, we extended MRAN from radial basis functions to elliptical basis functions. The resulting network is called MRAN-EBF. MRAN-EBF is applied to speaker verification and compared with MRAN.

Experimental results show that the performance of MRAN-EBF with full covariance is worse than that of MRAN. MRAN-EBF with diagonal covariance matrix performs slightly worse than MRAN in terms of false accept rate and false reject rate but has a higher complexity. One of the possible reasons why MRAN-EBF is unable to perform better MRAN is that the EKF algorithm used in MRAN-

EBF produces significant errors because of the non-linearity of the problem and huge number of independent parameters.

6.2 Recommendations for Further Research

Two possible directions of future works emerge from this thesis. One is about the applications of MRAN on other areas and the other is concerning the improvement of the MRAN and MRAN-EBF algorithms.

(1) Algorithm

- MRAN and MRAN-EBF use the Extended Kalman filter (EKF) to update network parameters during training. EKF is very computationally intensive. Although it has the advantage of fast convergence of network parameters, it could produce significant errors for highly non-linear problems. The unscented filter [40] can be used to replace EKF. Compared with EKF, the unscented filter is able to predict the state of highly non-linear system more accurately. Further more, it is much easier to implement [40].
- Recurrent radial basis function network has more compact networks compared to RBF networks [39]. The MRAN algorithm could be modified to train recurrent radial basis function networks, which may further reduce the network size.

(2) Application

- Radial basis function network has more compact network architecture compared with many other artificial neural networks. With the flexibility of adding and removing neurons dynamically during training, MRAN results even compact network. MRAN can be used on other types of biometric

applications, such as fingerprint identification, face recognition, etc.

Author's Publications

(1) Journal Paper:

- Li Guojie, P. Saratchandran and N. Sundararajan, “Text-Independent Speaker Verification Using Minimal Resource Allocation Networks”, accepted by *International Journal of Neural Systems*, 2004.

(2) Conference Paper:

- Li Guojie, P. Saratchandran and N. Sundararajan, “Speaker Verification Using Minimal Resource Allocation Networks”, *IEEE Intelligent Automation Conference*, Hong Kong, December 15-17, 2003.

Bibliography

- [1] A. G. Bors and M. Gabbouj, “Minimal topology for a radial basis functions neural network for pattern classification”, *Digital Signal Processing*, vol.4, pp.173-188, 1994.
- [2] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, 1970.
- [3] B. Atal, “Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification”, *J. Acoust. Soc. Am.*, 55(6), pp.1304-1312, 1974.
- [4] B. Poritz, “Linear predictive hidden Markov models and the speech signal”, *Proc. ICASSP 1982*, pp.1291-1294, 1976
- [5] C. Griffin, T. Matsui, and S. Furui, “Distance measures for text-independent speaker recognition based on MAR model”, *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Processing*, pp.309-312, 1994.
- [6] C. Montacie, P. Deleglise, F. Bimbot and J. Caraty, “Cinematic techniques for speech processing: temporal decomposition and multivariate linear prediction”, *Proc. IEEE Intl. Conf. on Acoustics,*

Speech, and Signal Processing, pp. I-153-156, 1992.

- [7] D.S. Broomhead and D.Lowe, “Multivariable functional interpolation and adaptive networks”, *Complex Systems* 2, pp.321-355, 1988.
- [8] F. Bimbot, L. Mathan, A. D. Lima and G. Chollet, “Standard and target driven AR-vector models for speech analysis and speaker recognition”, *IEEE-ICASSP*, vol.2, pp. II.5-II.8, San Francisco, USA, 1992.
- [9] F. K. Soong, A. E. Rosenberg, L. R. Rabiner and B. H. Juang, “A vector quantization approach to speaker recognition”, *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 387-390, 1985.
- [10] G. Doddington, M. Przybocki, A. Martin and D. Reynolds, “The NIST speaker recognition evaluation – Overview, methodology, systems, results, perspective”, *Speech Communication*, vol.31, pp.225-254, 2000.
- [11] G. H. Golub and C. F. Van Loan, *Matrix Computation*, Third Edition, Johns Hopkins, 1996.
- [12] H. J. Kunzel, “Current approaches to forensic speaker recognition”, *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, pp.135-141, 1994.
- [13] H. J. Kushner, “Dynamical equations for optimum non-linear filtering”, *Journal of Differential Equations*, pp.179-190, 1967.
- [14] I. Davidson and A. Satyanarayana, “Speeding up K-means clustering by bootstrap averaging”, *IEEE Data Mining Workshop on Clustering*

Large Data Sets, 2002.

- [15] J. Hartigan, *Clustering Algorithms*, Wiley Publishing, 1975.
- [16] J. Oglesby and J. S. Mason, "Optimization of neural models for speaker identification", *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 261-264, 1990.
- [17] J. Oglesby and J. S. Mason, "Speaker recognition with a neural classifier", *IEE First Int. Conf. on Artificial Neural Nets*, pp.306-309, 1989.
- [18] J. Moody and C. Darken, "Fast learning in network of locally-tuned processing units", *Neural Computation*, vol.1, pp.281-294, 1989.
- [19] J. Platt, "A resource allocating network for function interpolation", *Neural Computation*, vol.3, pp.213-225, 1991.
- [20] K. Tao, "A closer look at the radial basis function (RBF) networks", *Conference Record of 27th Asilomar Conference on Signals, System and Computers*, (CA, US), pp.401-405, 1993.
- [21] L. C. Jain and A. M. Fanelli, *Recent Advances in Artificial Neural Networks: Design and Applications*, CRC Press, 2000.
- [22] L. R. Bahl, P.F. Brown, P. V. de Souza, and R. L. Mercer, "Maximum mutual information estimation of hidden Markov model parameters", *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*,

pp.49-52, 1986.

- [23] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models", *IEEE ASSP Magazine*, pp.4-16, 1986.
- [24] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Inc., 1978.
- [25] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks", *Neural Computation*, vol.9, pp.461-478, 1997.
- [26] L. Yingwei, N. Sundararajan, and P. Saratchandran, "Performance evaluation of a sequential minimal radial basis function neural network learning algorithm", *IEEE Transactions on Neural Networks*, vol.9, pp.308-318, 1998.
- [27] M. Powell, "Radial basis functions for multivariable interpolation: a review", *Algorithms for Approximation*, pp.143-167, Oxford, 1987.
- [28] M. Savic and S. K. Gupta, "Variable parameter speaker verification system based on hidden Markov modeling", *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 281-284, 1990.
- [29] M. W. Mak and S. Y. Kung, "Estimation of Elliptical Basis Function Parameters by the EM Algorithm with Application to Speaker Verification", *IEEE Transactions on Neural Networks*, vol.11, pp.961-

969, July 2000.

- [30] N. J. Gordon, D. J. Salmond and A. F. M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”, *IEE Proceedings*, pp.107-113, 1993.
- [31] N. Sundararajan, P. Saratchandran and Y. W. Lu, *Radial Basis Function Neural Networks with Sequential Learning*, World Scientific, 1999.
- [32] N. Z. Tishby, “On the application of mixture AR hidden Markov models to text independent speaker recognition”, *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol.ASSP-30, pp.563-570, 1991.
- [33] P. Whittle, “On the fitting of multivariate autoregression and the approximate canonical factorization of a spectral density matrix”, *Biometrika*, vol.50, pp129-134, 1963.
- [34] R. Klevans and R. D. Rodman, *Voice Recognition*, Artech House, Inc., 1997.
- [35] S. Chen, C.F. Cowman, and P. Grant, “Orthogonal least squares learning algorithm for radial basis function networks”, *IEEE Transaction on Neural Networks*, vol. 2, pp.302-309, 1991.
- [36] S. Furui “An overview of speaker recognition technology”, *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, pp.1-9, 1994.

-
- [37] S. Furui, "Speaker-dependent-feature extraction, recognition and processing techniques", *Speech Communication*, vol.10, No.5-6, pp.505-520, 1991.
- [38] S. Furui, F. Itakura and S. Saito, "Talker recognition by longtime averaged speech spectrum", *Trans. IECE*, 55-A, Vol.1, No.10, pp.549-556, 1972.
- [39] S. Haykin, *Neural Networks – A Comprehensive Foundation*, Prentice Hall, 1999.
- [40] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," *SPIE AeroSense Symposium*, 1997.
- [41] S. Lee and R. Kil, "A Gaussian potential function network with hierarchically self-organizing learning", *Neural Networks*, vol.4, pp.207-224, 1991.
- [42] T. Artieres, Y. Bennani, P. Gallinari and C. Montacie, "Connectionist and conventional models for free-text talker identification tasks", *Neuronimes*, France, 1991.
- [43] T. Matsui and S. Furui, "Concatenated phoneme models for text-variable speaker recognition", *Proc. IEEE Intl. Conf on Acoustics, Speech, and Signal Processing*, pp.II-391-394, 1993.
- [44] T. Poggio and F. Girosi, "Networks for approximation and learning", *Proceedings of the IEEE*, vol.78, pp.1481-1497, 1988.

- [45] V. Kadiramanathan and M. Niranjan, "A function estimation approach to sequential learning with neural network", *Neural Computation*, vol.5, pp.954-975, 1993.
- [46] V. Kadiramanathan and M. Niranjan, "Application of an architecturally dynamic network for speech pattern classification", *Proceedings of the Institute Acoustics*, vol.14, pp.343-350, 1992.
- [47] W. M. Fisher, G. R. Doddington and K. M. Goudie-Marshall, "The DARPA speech recognition research database: specifications and status", *Proc. DARPA Workshop on Speech Recognition*, pp.93-99, 1986.
- [48] Y. H. Cheng and C. S. Lin, "A learning algorithm for radial basis function networks: with the capability of adding and pruning neurons", *IEEE Int. Conf. on Neural Networks*, vol.1, pp.797-801, 1994.
- [49] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantization", *IEEE Trans. On Communications*, vol.COM-28, No.1, pp.84-95, 1980.