Training Algorithm for Radial Basis Function Classifier

by

YILONG HAO

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2015
Acknowledgements

I would like to express my sincere gratitude to my advising professor Dr. Manry for his support in completing my research. I also appreciate him giving me financial support throughout my Master’s program. I am very grateful for his patience.

I would like to say thank you to my lab mates, especially Rohit and Kanishka who worked with me to debug and explain problems. Thank you Gautam, Son, Parastoo and Audy, who always let me feel the warm and harmonious environment of our lab. Everyone in our lab always helps each other without judgment. Thank you to everybody who gave me fond memories during my Master program.

August 13, 2015
Abstract

Training Algorithm for Radial Basis Function Classifier

Yilong Hao, M.S.

The University of Texas at Arlington, 2015

Supervising Professor: Michael T Manry

The computational complexity of kernel machines and their poor performance in the multi-label classification case is a major bottleneck to their success. In this thesis we present a systematic two step batch approach for constructing and training a new multiclass kernel machine (MKM). Unlike other kernel learning algorithms, the proposed paradigm prunes the kernels, and uses Newton’s method to improve the kernel parameters. In each iteration, output weights are found using orthogonal least squares. The proposed hybrid training algorithm is compared with least square support vector machines (LS-SVM) and support vector machines (SVM). Simulations results on many benchmark and real life datasets show that the proposed algorithm has significantly improved convergence speed, network size and generalization over conventional kernel machine training algorithms.
# Table of Contents

Acknowledgements ................................................................................................................ iii  
Abstract................................................................................................................................ iv  
List of Illustrations ................................................................................................................. vii  
List of Tables ....................................................................................................................... viii  
List of Symbols ...................................................................................................................... ix  
List of Acronyms .................................................................................................................... xi  

Chapter 1 Introduction ........................................................................................................... 1  
  1.1 Neural Network ........................................................................................................ 1  
  1.2 Benefits of Neural Networks .................................................................................... 2  
  1.3 Common Types of Neural Networks ........................................................................ 3  
    1.3.1 Multi-layer Perceptron (MLP) ........................................................................ 4  
    1.3.2 Radial Basis Function Networks .................................................................. 4  
  1.4 Kernel Machines ...................................................................................................... 5  
  1.5 Research Motivation ................................................................................................ 5  
  1.6 Organization of the thesis ........................................................................................ 6  

Chapter 2 RBF Neural Network Review ................................................................................ 7  
  2.1 Training Data ............................................................................................................ 7  
  2.2 RBF structure and operation .................................................................................... 7  
  2.3 RBF Neural Network parameter initialization ........................................................... 9  
  2.4 Optimal Output Weights ......................................................................................... 10  
  2.5 Pruning Method ....................................................................................................... 11  
    2.5.1 Ordered Basis Function[30][28] .................................................................. 11  
    2.5.2 Validation Error ........................................................................................... 13  

Chapter 3 Common Kernel Methods ................................................................................... 15  
  3.1 Support Vector Machines ....................................................................................... 15  
  3.2 Mathematical treatment of SVMs........................................................................... 15  
  3.3 Problems with SVMs .............................................................................................. 18  

Chapter 4 Least Square Support Vector Machines............................................................. 19
4.1 LS-SVMs for binary classification [43] ................................................................. 19
4.2 Multiclass classification for the LS-SVM .............................................................. 20
4.3 Problems with LS-SVM ....................................................................................... 20
Chapter 5 Optimization for the RBF Neural Network .................................................. 22
  5.1 One Pass Validation ......................................................................................... 22
  5.2 Optimize Spread Parameter with Newton’s Method ......................................... 22
  5.3 Regularization .................................................................................................. 23
  5.4 Output Reset (OR) ............................................................................................ 24
  5.5 MKM Training Algorithm .................................................................................. 25
Chapter 6 Simulation Results ....................................................................................... 26
Chapter 7 Conclusion and Future Work ..................................................................... 29
Appendix A Description of Data Sets Used For Training, Validation and Testing ........ 30
Appendix B Schmidt procedure .................................................................................. 38
Appendix C Performance Evaluation of RBF Neural Network .................................... 43
Appendix D K-fold Cross Validation ......................................................................... 45
References ................................................................................................................. 47
Biographical Information ......................................................................................... 61
List of Illustrations

Figure 1-1 A simple nonlinear neuron ............................................................................................ 1

Figure 1-2 Multilayer perceptron with one hidden layer ................................................................. 4

Figure 2-1 Structure of RBF Neural Network with bias .................................................................... 8

Figure 3-1 Structure of support vector machine ............................................................................. 16
List of Tables

Table 6-1 Specification of datasets .................................................................................................. 33
Table 6-2 Testing result after implement each techniques ................................................................. 33
Table 6-3 Testing Performance comparison of MKM, LS-SVM, and SVM ..................................... 34
Table 6-4 Number of hidden units for MKM comparison of dataset............................................. 34
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>Input vector</td>
</tr>
<tr>
<td>$X_p$</td>
<td>$p^{th}$ basis function</td>
</tr>
<tr>
<td>$t$</td>
<td>Desired output vector</td>
</tr>
<tr>
<td>$t_p$</td>
<td>$p^{th}$ desired output vector</td>
</tr>
<tr>
<td>$t_p(i)$</td>
<td>$i^{th}$ element of $t_p$</td>
</tr>
<tr>
<td>$y_p$</td>
<td>actual output vector</td>
</tr>
<tr>
<td>$y_p(i)$</td>
<td>$i^{th}$ element of $y_p$</td>
</tr>
<tr>
<td>$i_c$</td>
<td>correct class vector</td>
</tr>
<tr>
<td>$i_c(p)$</td>
<td>correct class number for $p^{th}$ pattern</td>
</tr>
<tr>
<td>$P$</td>
<td>row number in the data file</td>
</tr>
<tr>
<td>$N_v$</td>
<td>number of rows in data file</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of inputs</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of outputs</td>
</tr>
<tr>
<td>$N_h$</td>
<td>Number of hidden units</td>
</tr>
<tr>
<td>$N_c$</td>
<td>number of classes</td>
</tr>
<tr>
<td>$W$</td>
<td>output weight matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>Constant</td>
</tr>
<tr>
<td>$w(i,k)$</td>
<td>weight from the $k^{th}$ hidden unit to the $i^{th}$ output</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$m_k$</td>
<td>$k^{th}$ center vector</td>
</tr>
<tr>
<td>$R$</td>
<td>auto correlation matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>cross correlation matrix</td>
</tr>
<tr>
<td>$\beta_k$</td>
<td>spread parameter of the $k^{th}$ hidden unit</td>
</tr>
</tbody>
</table>
List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>Multi-Layer Perceptron</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>OR</td>
<td>Output Reset</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>LS-SVM</td>
<td>Least Square Support Vector Machine</td>
</tr>
</tbody>
</table>
An artificial neural network (ANN) is an information processing model inspired by biological nervous systems, such as the brain. The key element of the model is the novel structure, it is composed of a huge number of highly interconnected processing nodes each meant for solving specific problems. ANNs can learn from examples like a human. In a nonlinear neuron, a structure called the synapse connects two nodes and each synapse has an appropriate value called the synaptic weight. The synaptic weights are multiplied by the input signal at the head of the synapse to obtain an output at the end of the synapse. There is a summing junction called the net value of the neuron, which sums up the outputs of all the synapses connected to it. The final output of a neuron can be obtained by the net value through an activation function.

Figure 1-1  A simple nonlinear neuron
Artificial neural networks have been used in many fields, such as image processing\([1][67][68][69][70][71][75]\), control systems\([2][76][77][78][79][80][81]\), medical image analysis\([3][82][83][84][85]\), prediction (such as predicting Stock Index or cancers)\([58][61][65][74][86][87][88]\) and pattern recognition (such as face detection and finger detection) \([4][72][73][89][90][91][92][93][94][95]\). There are many applications of neural networks in present day businesses\([5][58][59][60][61][62][105]\). Financial institutions are developing superior neural network models for credit card risk and bankruptcy\([58][59][60][96][97][98][99]\). Neural networks are used to forecast stock market prices\([62][100][101][102][103][104][105]\). Oil and gas corporations are learning more from their data by using neural networks to predict oil PVT\(\text{(pressure-volume-temperature)}\)\([64][106][107][108][109]\). In the medical area, neural networks technology can be used for classification and diagnostic prediction of cancers\([65][110][111][112][113][114][115]\).

1.2 Benefits of Neural Networks

Neural networks have the following useful properties and capabilities\([9][10]\):

1. Nonlinearity: An artificial neuron is nonlinear because of its activation function. A neural network made up of such elements is also nonlinear. This property is extremely important, especially when modeling nonlinear phenomenon \([11, 12]\)

2. Input-Output Mapping: In supervised learning, the synaptic weights of a neural network can be modified to reduce the error between the desired output and the actual output. The training of the network is repeated for many patterns, until we can ignore the changes in the synaptic weights. Thus the network learns from the training data by constructing an input-output
mapping[9]. So they are useful in regression analysis, such as time series prediction, fitness approximation and modeling [13].

3. Adaptivity: Neural networks have the capability to adapt their synaptic weights to changes in the surrounding environment. Particularly, a neural network trained to operate in a specific environment can easily be retrained to deal with minor changes in the operating environmental conditions[9].

4. Evidential Response: In the context of pattern recognition, a neural network based classifiers can be designed to provide information not only about the predicted class of a pattern, but also about confidence in the decision made[9]. This helps in eliminating ambiguous patterns.

5. Contextual Information: Every neuron in the network is potentially affected by the activity of all other neurons in the network. So, contextual information is dealt with naturally by neural network[9].

6. Due to the highly distributed information stored in a neural network, the loss of or damage to one neuron does not affect the performance of the whole network drastically[10]. There is a graceful degradation in performance [14].

1.3 Common Types of Neural Networks

The well-known neural networks are the multi-layer perceptron (MLP)[116] and radial basis function neural (RBF) network[25].
1.3.1 Multi-layer Perceptron (MLP)

The MLP consists of multiple layers of computational units, usually interconnected in a feed forward way. The MLP has one or more hidden layers between the input layer and the output layer. Usually, each neuron in one layer connects to all the neurons of the following layer.

![Multilayer perceptron with one hidden layer](image)

Figure 1-2  Multilayer perceptron with one hidden layer

1.3.2. Radial Basis Function Networks

An RBF neural network is a three-layer feed forward network consisting of a single hidden layer which uses radial basis activation functions (such as the Gaussian function)[9].
The RBF neural network has neurons with nonlinear RBF activations in the hidden layer and linear summation activation functions in the output layer.

1.4 Kernel Machines

Kernel methods owe their name to the use of kernel functions, which enable them to map a low-dimension space to a high-dimension space[118]. The main idea is[119]: the integral point set which cannot be linearly segmented in a low dimensional space can be linearly segmented if it is transformed to a high-dimensional space. In machine learning, kernel methods are a class of algorithms for pattern analysis[120].

The most well-known kernel machine is the support vector machine (SVM) which can be used for classification and regression analysis[15][42][118][121][122].

1.5 Research Motivation

SVMs are widely used in binary classification due to their simplicity of implementation. However, they can also be used in regression problems and multi-class classification problems. Although the SVM and LS-SVM [40][41][42][43][44] are available for multi-class classification problems, available SVM and LS-SVM software[123] tools are complicated to use for entry level users. The network parameters are not easy to select properly for good performance and the training algorithms are not easily programmed and modified. Newton’s method is not used to optimize the kernel parameters. The number of support vectors can be extremely large, which results in the high computation[53]. Even though both the SVM and LS-SVM can solve multiclass classification problems, their performance for multiclass classification is not good enough[52][56][117]. Hence a different approach for designing kernel machines is needed.
1.6 Organization of the thesis

In this thesis, we develop a training method for RBF based kernel machines for both binary and multi-class classification. Chapter 2 reviews the structure, notation and training of conventional RBF networks. Chapter 3 reviews the support vector machine algorithm and its training. In chapter 4, we introduce a training method for the least square support vector machine (LS-SVM). Finally, we discuss the SVM’s problems. In chapter 5, we improve RBF training using a pruning method [28], Newton’s method, regularization [31], and output reset method [57]. In chapter 6, we focus on the simulation results on several widely available data files, and we make comparisons with the SVM and LS-SVM training algorithms. In chapter 7, we present our conclusions and possible enhancements to this work.
Chapter 2

RBF Neural Network Review

2.1 Training Data

The training data is a set of data consisting of input vectors and label vectors. The training dataset consists of \( N_v \) training patterns \( \{x_p, i_c(p)\} \), \( 1 \leq p \leq N_v \), where \( x_p \) is the \( p \)th input vector with dimension \( N \), and \( i_c(p) \) is the \( p \)th desired output class label. \( i_c(p) \) is between 1 and \( M \), where \( M \) is the total number of classes. The class label \( i_c(p) \) is converted to a desired output vector \( t_p \) with dimension \( M \) as

\[
t_p(i) = \delta(i - i_c(p)) \quad 1 \leq i \leq M
\]

\( x_p(n) \) denotes the \( n \)th element of \( x_p \). So \( x_p \) can be represented as

\[
x_p = [x_p(1), x_p(2), ..., x_p(N)]^T
\]

\( t_p(i) \) denotes the \( i \)th desired output for the \( p \)th input pattern. \( y_p \) denotes the actual output vector when \( x = x_p \), so \( y_p(i) \) is the \( i \)th element of \( y_p \). \( N_h \) is the number of hidden units. \( W \) is the output weight matrix and \( w(i, k) \) is the output weight from the \( k \)th hidden unit to the \( i \)th output unit.

2.2 RBF structure and operation

The RBF neural network is a three layer feed forward neural network which has a single hidden layer that uses radial basis activation functions (such as the Gaussian function) \([9]\). Its layers are the input layer, hidden layer, and output layer. Each neuron in the input layer connects to each neuron in the hidden layer, and each neuron in the hidden layer consists of a radial basis function (e.g. Gaussian). The output layer has a weighted sum of outputs from the hidden layer to form the network outputs.
For mapping $x_p$ to $y_p$, we have the following steps

1. Initially we have $N_v$ center vector $m_k$, which are equal to the input training vectors $x_k$, where $k$ varies from 1 to $N_v$. $\beta_k$ is defined as the spread parameter, $\mu_k$ is defined as the mean value of the elements of $m_k$. For the $p^{th}$ training pattern, $d(x_p, m_k)$ is defined as the 2-norm distance between $x_p$ and $m_k$[25]:

$$d(x_p, m_k) = \sum_{n=1}^{N} (x_p(n) - m_k(n))^2 \quad (2.2)$$

2. Initially, the hidden layer consists of the basis vector $X_p$ of length $N_h + 1$, where $X_p(1) = 1$, and the remaining elements of $X_p$ are calculated from $x_p$, the spread parameter $\beta_k$, and the center vector $m_k$ as:
\[ X_p(k + 1) = \exp(-\beta_k d(x_p, m_k)) \quad k = 1, 2, \ldots N_h \]  

(2.3)

(3) Calculate auto correlation matrix \( R \) and cross correlation matrix \( C \)

(4) Calculate weight matrix \( W \) by using \( R \) and \( C \)

(5) The hidden layer is fully connected to the output layer via output weights. The weights which connect from each hidden unit to each output unit form a \( M \times (N_h + 1) \) weight matrix \( W \), and \( y_p \) is calculated as:

\[ y_p(i) = \sum_{k=1}^{Nh+1} w(i, k)X_p(k) \]  

(2.4)

or

\[ y_p = W \cdot X_p \]  

(2.5)

2.3 RBF Neural Network parameter initialization

Given the training data \( \{x_p, t_p\} \), initialize the center vectors as \( m_k = x_k \) for \( k \) between 1 to \( N_v \).

We initialize \( \beta_k \) as[9]

\[ \beta_k = 1/\left(2 \sum_{n=1}^{N} (m_k(n) - \mu_k)^2 \right) \]  

(2.6)

where

\[ \mu_k = \frac{1}{N} \sum_{n=1}^{N} m_k(n) \]  

(2.7)

The error function of an RBF is measured using the Mean Square Error (MSE) as:

\[ E = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{M} [t_p(i) - y_p(i)]^2 \]  

(2.8)
2.4 Optimal Output Weights

We consider a linear system mapping an \((N_h + 1)\) dimensional hidden layer basis vector \(X_p\) to an \(M\) dimensional output vector \(y_p\). \(X_p\) is obtained by equation (2.3). The \((N_h + 1) \times (N_h + 1)\) auto correlation matrix \(R\) is defined as:

\[
r(k,n) = \frac{1}{N_v} \sum_{p=1}^{N_v} X_p(k) \cdot X_p(n)
\]  

(2.9)

The \((N_h + 1) \times M\) cross-correlation matrix \(C\) is defined as:

\[
c(k,i) = \frac{1}{N_v} \sum_{p=1}^{N_v} X_p(k) \cdot t_p(i)
\]  

(2.10)

The weight matrix \(W\) can be solved by the following steps:

1. Writing equation (2.8) in terms of elements of \(W\):

\[
E = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{M} \left[ t_p(i) - \sum_{k=1}^{N_h+1} w(i,k) \cdot X_p(k) \right]^2
\]  

(2.11)

Differentiating \(E\) with respect to the elements of \(W\):

\[
\frac{\partial E}{\partial w(m,n)} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \left( t_p(m) - \sum_{k=1}^{N_h+1} w(m,k) \cdot X_p(k) \right) \cdot X_p(n)
\]  

(2.12)

Then we get:

\[
\frac{\partial E}{\partial w(m,n)} = \sum_{p=1}^{N_v} \left( t_p(m)X_p(n) - \sum_{k=1}^{N_h+1} w(m,k) \cdot X_p(k)X_p(n) \right)
\]  

(2.13)

\[
= \sum_{p=1}^{N_v} \left( t_p(m)X_p(n) - \sum_{k=1}^{N_h+1} w(m,k) \cdot \sum_{p=1}^{N_v} X_p(k)X_p(n) \right)
\]  

(2.14)

Writing equation (2.13) in terms of the auto correlation and cross correlation matrices we have:

\[
\frac{\partial E}{\partial w(m,n)} = \sum_{p=1}^{N_v} \left( c(n,m) - \sum_{k=1}^{N_h+1} w(m,k) \cdot r(k,n) \right)
\]  

(2.15)
For minimizing the mean square error, let the partial derivatives be zero. Then the equations can be represented in a compact way as:

\[ R \cdot W^T = C \]  

(2.16)

2.5 Pruning Method

Initially, the center vectors \( m_k \) are initialized as equal to the input vectors as \( m_k = x_k \), where \( k \) varies from 1 to \( N_h \), and \( N_h = N_v \). Thus, the number of the center vectors are very large since \( N_v \) is usually large. In [28], the orthogonal least squares (OLS) method is employed as a forward regression procedure [29] to select a suitable set of center vectors from a large set of candidates. The procedure chooses basis functions one by one till an adequate network has been chosen based on the analysis of residuals. In this thesis, we use the OLS developed in [31] which acts on the correlation matrices. We choose the useful hidden units with a unique one pass pruning-with-validation method which uses OLS [28][31].

2.5.1 Ordered Basis Function [30][28]

The aim of pruning is to eliminate the less useful hidden units, and keep the useful hidden units which have information relevant for estimating outputs. Here, we use OLS [27][28] to eliminate the less useful hidden units including those which are linearly dependent upon others.

The algorithm first optimally orders hidden units on the basis of their usefulness. We get the orthonormal basis functions by the Schmidt procedure.

Let \( o(m) \) be the optimal order in terms of usefulness of the hidden units, so that \( o(m) \) specifies the order in which raw basis function \( X_k \) will be processed into orthonormal basis function \( X'_1 \).

\( N_u \) is equal to \( N_h + 1 \). For ordered basis function, we get the \( m^{th} \) orthonormal basis function as, [28][30]
$X'_m = \sum_{k=1}^{m} a_{mk}X_{o(k)}$  \hspace{1cm} (2.17)

Initially, $X'_1$ is found as $a_{11}X_{o(1)}$ where,

$$a_{11} = \frac{1}{\|X_{o(1)}\|} = \frac{1}{[r(o(1), o(1))]^2}$$  \hspace{1cm} (2.18)

We get

$$c_i = \sum_{k=1}^{i} a_{ik} r(o(k), o(m)) \quad \text{for} \quad 2 \leq m \leq N_u$$  \hspace{1cm} (2.19)

We set $b_{m} = 1$, then we have

$$b_k = - \sum_{i=k}^{m-1} c_ia_{ik} \quad \text{for} \quad 1 \leq i \leq m - 1$$  \hspace{1cm} (2.20)

$$a_{mk} = b_k \left[ r(o(m), o(m)) - \sum_{i=1}^{m-1} c_i^2 \right]^{\frac{1}{2}} \quad \text{for} \quad 1 \leq k \leq m - 1$$  \hspace{1cm} (2.21)

$$w_o'(i, m) = \sum_{k=1}^{m} a_{mk} c(i, o(k)) \quad 1 \leq k \leq m$$  \hspace{1cm} (2.22)

where $W_o'$ are the weights for the ordered basis functions. In order to find the most useful basis function, we treat each basis function $X_k$ as if it were the first one, find $w_o'(i, m)$ for all $i$, and sum up squares of the $w_o'(i, m)$, over $i$. If $m_o$ denotes the value of $m$ yielding the largest sum, then $X_{m_o}$ is the most useful basis function.

The selection process will be used to optimally order the hidden units $N_h + 1$. We now define notation to help us specify the candidate set of basis function to choose during ordering.

First define $S(m)$ as the set of indices of chosen basis functions where $m$ is the number of hidden units. Then $S(m)$ is given by

$$S(m) = \begin{cases} \emptyset & \text{for} \quad m = 0 \\ \{o(1), o(2), \ldots, o(m)\} & \text{for} \quad 0 < m \leq N_h + 1 \end{cases}$$  \hspace{1cm} (2.23)
Let’s take $o(1) = 1$, putting the threshold as a first hidden unit. The set of candidate basis functions is clearly $S_c(m) = \{1, 2, \ldots, N_h + 1\} - S(1)$, so we get $S_c(m)$ as:

$$S_c(m) = \{2, \ldots, N_h + 1\}$$  \hspace{1cm} (2.24)

For $2 < m < N_h + 1$, we get $S_c(m - 1)$. For each trial value of $o(m)$ belonging to $S_c(m - 1)$, perform operations of equation (2.19),(2.20),(2.21),(2.22). We define $P(m)$ is:

$$P(m) = \sum_{i=1}^{M} w_o'(i, m)^2$$  \hspace{1cm} (2.25)

The trial value of $o(m)$ which maximizes $P(m)$ is found. Assuming that $P(m)$ is maximum when testing the $k^{th}$ element, then $o(m)=k$. $S(m)$ is updated as

$$S(m) = S(m - 1) \cup \{o(m)\}$$  \hspace{1cm} (2.26)

Then for the general case the candidate basis functions are

$$S_c(m - 1) = \{1, 2, \ldots, N_h + 1\} - \{o(1), o(2), \ldots, o(m - 1)\}$$  \hspace{1cm} (2.27)

By using equation (2.27) after testing all the candidate basis function, $o(m)$ takes its value and $S(m)$ is updated according to equation(2.25). The process is repeated until $m=N_h + 1$, after the complete $o(m)$ function is obtained, both the original basis functions and the orthonormal ones are ordered. Then the orthonormal weights are mapped to normal weights.

### 2.5.2 Validation Error

We use validation data to implement the pruning. Let $y_p(i, m)$ represent the $i^{th}$ output of the network which is having $m$ hidden units for the $p^{th}$ pattern, let $N_v(m)$ represent the misclassified validation patterns with $m$ hidden units. Let $X_p'(m)$ represent the $m^{th}$ ordered orthonormal basis function. For $1 \leq i \leq M$ and $1 \leq m \leq N_h + 1$, $y_p(i, m)$ is calculated as

$$y_p(i, 1) = w'(i, 1) \cdot X_p'(1)$$  \hspace{1cm} (2.28)
\[ y_p(i, m) = y_p(i, m - 1) + w'(i, m) \cdot X_p'(m) \]  
(2.29)

Define \( i'_c(m) = \arg\max_{1 \leq i \leq M} (y_p(i, m)) \).

Then \( N_e(m) \) is calculated as

\[ N_e(m) \leftarrow N_e(m) + \left( 1 - \delta(i'_c - i_c(p)) \right) \]  
(2.30)

Let \( P_{ev} \) represent the misclassification for the validation data with \( N_h + 1 \) hidden units, where

\[ P_{ev}(m) = \frac{N_e(m)}{N_v} \]  
(2.31)

\( P_{ev} \) is calculated efficiently in one pass through the validation data. Define \( N_{hd} \) as the best number of hidden units. Compute the validation error using the ordered basis functions and validation data and find the \( N_{hd}^{th} \) hidden unit which gives the minimum \( P_{ev} \). So the first \( N_{hd} \) hidden units are kept, the remaining units are pruned by deleting the last \( N_h + 1 - N_{hd} \) hidden units.
Chapter 3

Common Kernel Methods

In this chapter, we will review support vector machine and its training algorithms.

3.1 Support Vector Machines

Support Vector Machines (SVMs)[36][37][38][39] are the most well-known learning systems based on kernel methods for solving pattern recognition problems. They have been shown to be effective for many classification problems[16][17][18][19]. In the method, the SVM maps the data into a higher dimensional input space and constructs an optimal separating hyperplane between the positive and negative classes with the maximal margin in the space.

3.2 Mathematical treatment of SVMs

Support vector machines have one output but two classes. The scalar output \( y_p \) is calculated as [21]

\[
y_p = w^T \cdot x_p - b
\]

(3.1)

where the coefficient vector \( w \) is \( N_{sv} \) by 1 and \( b \) is a bias. The basis vector \( x_p \) is \( N_{sv} \) by 1, and \( 1 \leq p \leq N_{sv} \). Assume support vectors make up the first \( N_{sv} \) patterns, so \( x_p \) is a support vector of dimension \( N_{sv} \). The vector \( x_p \) is generated from the \( N \) by 1 input vector \( x_p \) as in the MLP, but the activation function is different.

The support vectors are a subset of the basis vectors \( x_p \) for which \( y_p = t_p \).
Consider the binary classification task, where \( N_c = 2 \). We have a training set \( \{ \mathbf{x}_p, t_p \} \), \( p=1,2, ..., N_v \). The equation of a decision surface in the form of a hyperplane is[25]:

\[
\mathbf{w}^T \cdot \mathbf{X}_p - b = 0
\]  \hspace{1cm} (3.2)

where coefficient vector \( \mathbf{w} \) is \( N_{sv} \) by 1, and \( b \) is a bias. Note that basis vectors \( \mathbf{X}_p \) which are not support vectors satisfy \( (N_{sv} + 1) \leq p \leq N_v \).

For a given weight vector \( \mathbf{w} \) and bias \( b \), the separation between the hyperplane and the support vector is called the margin of separation[9]. Clearly, there are many possible separating hyperplanes. The goal of a support vector machine is to find the particular hyperplane for which the margin of separation is maximized.

From Vapnik's statistical learning theory[23], for \( 1 \leq p \leq N_{sv} \)

\[
\mathbf{w}^T \cdot \mathbf{X}_p - b \geq 1 \quad \text{if} \quad t_p = 1
\]  \hspace{1cm} (3.3)
\[ w^T \cdot x_p - b \leq -1 \quad \text{if } t_p = -1 \] (3.4)

The margin of separation between the upper bound and the lower bound is[22]:

\[ \frac{2}{\|w\|} = \frac{2}{\sqrt{w^T w}} \] (3.5)

So, if we want to maximize the margin of separation \( \frac{2}{\|w\|} \), it is equivalent to minimizing \( \|w\|^2 \).

The primal problem is[22]

\[
\min E_{svm} = \frac{1}{2} \|w\|^2 + C \sum_{p=1}^{N_v} \xi_p
\] (3.6)

where \( C \) is a user- specified positive parameter, and the \( \xi_p \) are called the slack variables. We need to find the optimum values of the weight vector \( w \), bias \( b \) and the slack variables \( \xi_p \) to minimize \( E_{svm} \). We can formulate the dual problem as follow:

Based on the Karush-Kuhn-Tucker theorem [15], we have the dual form for the constrained optimization of a support vector machine as:

Given the training data \{ \( x_p, t_p \), \( p=1,2, \ldots, N_v \), find the Lagrange multipliers \( \alpha_p \) which maximize the objective function

\[
y_p = \sum_{p=1}^{N_v} \alpha_p - \frac{1}{2} \sum_{p=1}^{N_v} \sum_{k=1}^{N_v} \alpha_p \alpha_k t_p t_k K(x_p, x_k)
\] (3.7)

where \( K(x_p, x_k) = \exp(-\|x_p - x_k\|^2/(2\sigma_k^2)) \)

subject to the constraints

\[
\sum_{p=1}^{N_v} \alpha_p y_p = 0
\] (3.8)

\[
0 \leq \alpha_p \leq C
\] (3.9)
where \( K(x_p, x_k) \) is the RBF kernel function \( X_p(k) \) where \( m_k = x_k \). Some common types of support vector machines\[52][128\] are polynomial learning machine and radial-basis-function network.

3.3 Problems with SVMs

SVMs are difficult for entry level users. The network parameters are not easy to select for good performance and the training algorithms are not easily programmed or modified. The number of support vectors can be extremely large, which results in a high computational load, and SVMs can take a long time to train. Even though SVMs can be used for multiclass classification problems, the results are often not satisfactory.
Chapter 4

Least Square Support Vector Machines

The Least Square Support Vector Machines (LS-SVMs) which is the least squares formulation of SVM, has been proposed [40][41], which involves the equality constraints only [42]. LS-SVMs are a set of supervised learning related methods which analyze data and recognize patterns, and which are used for classification and regression analysis.

4.1 LS-SVMs for binary classification [43]

Given a training set \{ x_p, p=1,2, ..., N_v \} and corresponding binary class labels \( t_p \in \{-1,+1\} \), Vapnik’s SVM classifier formulation was modified by Suykens[41] into the following LS-SVM formulation:

\[
\min \ E_{\text{lssvm}} = \frac{1}{2} \| \mathbf{w} \|^2 + \gamma \frac{1}{2} \sum_{p=1}^{N_v} (e_p^2) \tag{4.1}
\]

subject to the equality constraints instead of inequality constraints.

\[
y_p ( \mathbf{w}^T \mathbf{x}_p - b ) = 1 - e_p \tag{4.2}
\]

where \( \gamma \) is the regularization constant[66]. \( b \) is the output threshold parameter. \( e_p \) are called the slack variables[41]. Then, we construct the Lagrangian function with the Lagrange multipliers as.

\[
L(\mathbf{w}, b, \mathbf{e}, \alpha) = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{p=1}^{N_v} \alpha_p \{ y_p ( \mathbf{w}^T \mathbf{x}_p - b ) - 1 + e_p \} \tag{4.3}
\]

where \( \alpha_p \) are the Lagrange multipliers, \( \mathbf{e} \) are called the slack variables. Differentiating \( L(\mathbf{w}, b, \mathbf{e}, \alpha) \) with respect to \( \mathbf{w}, b, \mathbf{e} \) and \( \alpha \) and equating to zero yields[66]

\[
\mathbf{w} = \sum_{p=1}^{N_v} \alpha_p \mathbf{x}_p \tag{4.4}
\]
\[
\sum_{p=1}^{N_v} \alpha_p = 0 \tag{4.5}
\]
\[
\alpha_p = \gamma e_p \tag{4.6}
\]
\[
y_p(w^T x_p - b) - 1 + e_p = 0 \tag{4.7}
\]

After taking the conditions for optimality, the LS-SVM classifier with RBF kernel can be constructed as follows:
\[
y_p = \sum_{p=1}^{N_v} \alpha_p - \frac{1}{2} \sum_{p=1}^{N_v} \sum_{k=1}^{N_v} \alpha_p \alpha_k t_p t_k K(x_p, x_k) \tag{4.8}
\]

where
\[
K(x_p, x_k) = x_p(k) = \exp(-\|x_p - x_k\|^2/(2\sigma_k^2))
\]

4.2 Multiclass classification for the LS-SVM

Multiclass LS-SVMs have been proposed in [44]. The task of an M-class classifier is to predict the class label \( C_m, m = 1, \ldots, M \), given a new input vector \( x \). A popular way to solve the M class problem is to reformulate the problem into a set of L binary classification problems[44][45][46][47][48][49][50]. The first method is to construct M(M-1)/2 one-versus-one binary classifiers, each classifier discriminating between each pair of classes[47][51]. In an alternative approach [48], a minimal output coding(MOC), has been applied to solve the multiclass problem with binary least square support vector machines, using L bits to encode up to \( 2^L \) classes. The one-versus-all and error correcting output codes(ECOC) approach also can solve the multiclass problems in LS-SVM.

4.3 Problems with LS-SVM

The LS-SVM is difficult to use for entry level users. The network parameters are not easy to select properly for good performance and the training algorithms are not easily generated and
programmed. The drawback of the LS-SVM[124] is that sparseness is lost in the LS-SVM solution. In this case every data point is contributing to the model and the relative importance of a data point is given by its support value. So the number of support vectors can be extremely large, which results in the high computation, and LS-SVM can take a long time to train. Even though LS-SVM can be used for multiclass classification problems and gets better results than SVMs, the results are not good enough.
Chapter 5

Optimization for the RBF Neural Network

5.1 One Pass Validation

Our aim is to get a validation error $P_{ev}$ versus RBF hidden units which are the basis functions.

Calculating the error $P_{ev}$ over all hidden units from chapter 2, thus we generate the validation mean square error versus the hidden units size curve in one pass through the validation data set. Then, the pruning is completed with the following steps:

1. We need to find $N_{h, best}$, the numbers of hidden units which minimize the validation error.

2. We delete the ordered orthonormal basis functions after $N_{h, best}$, so $X'_k$ only keeps the first $N_{h, best}$ units. We then set $N_h = N_{h, best}$.

3. Find $X_k$ from the $X'_k$ and $A$ matrices as $X_k = A^{-1}X'_k$.

4. We rearrange RBF center vectors $m_k$ and $\beta_k$ in the same order, for $0 < k \leq N_h$.

5.2 Optimize Spread Parameter with Newton’s Method

The initial spread parameter cannot get good results. So after we prune the center vectors, we can optimize the spread parameters $\beta_k$. In chapter 2 we have initialized the spread parameters and defined them as the inverse of the standard deviation of $m_k$. We use the error function as in (2.8). For the $p^{th}$ pattern, the $k^{th}$ hidden unit output is:

$$X_p(k+1) = \exp(-\beta_k d(x_p, m_k))$$

(5.1)

The output vector $y_p$ is the same as described in (2.5).

We calculate the gradient for $\beta$ as[127]:

$$g_{\beta}(k) = -\frac{\partial E}{\partial \beta_k} = \frac{2}{N_p} \sum_{p=1}^{N_v} \sum_{i=1}^{M} \left[ t_{p(i)} - y_{p(i)} \right] \frac{\partial y_{p(i)}}{\partial \beta_k}$$

(5.2)

where
\[
\frac{\partial y_p(i)}{\partial \beta_k} = -\sum_{k=1}^{N_h+1} \sum_{n=1}^N w(i, k) \cdot X_p(k) \cdot (x_p(n) - m_k(n))^2
\]  

(5.3)

Combining (5.29) and (5.30), we get \( g_\beta \). Then we can get the Hessian matrix element as:

\[
h_\beta(u, v) = \frac{\partial^2 E}{\partial \beta_u \partial \beta_v} = 2 \sum_{p=1}^{N_v} \left( \sum_{i=1}^M \frac{\partial y_p(i)}{\partial \beta_u} \cdot \frac{\partial y_p(i)}{\partial \beta_v} \right)
\]

(5.4)

We get the following equation

\[
H_\beta \cdot z = g_\beta
\]

(5.5)

Solving by using OLS, we can get \( z \), \( \beta_k \) can be updated as

\[
\beta_k \leftarrow \beta_k + z_k
\]

(5.6)

5.3 Regularization

If too many hidden units are used, RBF neural network may result in poor performance because of overfitting[126]. One method to solve the overfitting problem is to use regularization [31] as in SVM design. We have the error function of (2.8), now we add a weight penalty to the error function as:

\[
E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \left[ t_p(i) - y_p(i) \right]^2 + \lambda \| w \|^2
\]

(5.7)

\[
= \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \left[ t_p(i) - \sum_{k=1}^{N_h+1} w(i, k) \cdot X_p(k) \right]^2 + \lambda \| w \|^2
\]

Differentiating \( E \) with respect to the elements of \( w \), and writing equation in terms of the auto correlation and cross correlation matrices we have:

\[
\frac{\partial E}{\partial w(m, n)} = -2 \frac{1}{N_v} \sum_{p=1}^{N_v} \left( c(n, m) - \sum_{k=1}^{N_h+1} w(m, k) \cdot r(k, n) \right) + 2 \lambda w(m, n)
\]

(5.8)

For minimizing the mean square error \( E \), we equate the derivative to be zero, then the equations can be represented in a compact way as:
\[ C - R \cdot W^T = \lambda W \quad (5.9) \]

or

\[ W = (R + \lambda I)^{-1} C \quad (5.10) \]

We then minimize \( E_v \) with respect to \( \lambda \), where \( E_v \) is the validation error using the regularization.

### 5.4 Output Reset (OR)

OR is an algorithm that we use to prevent distortion of class boundaries. Each of the individual outputs could perform better than expected or worse and some of them might still be memorized. As an example suppose we have uncoded outputs for a multiclass classifier. If instead of being 1 or -1, the correct class output is 1.5 which is better than expected for positive class, then In that case OR would not count that as an error. It would be the same case for a negative class, if the output is -1.5. However, for the incorrect class, if the output is 0.7, OR will not affect it and will count as error.

The output reset algorithm [57] [125] can minimize the training error.

\[ E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{M} \left[ t_p(i) - y_p(i) \right]^2 \quad (5.11) \]

In the basic OR algorithm [57] [125], first, we set the desired output equal to the actual output when the output has the correct sign but is larger than 1 in magnitude, or when the output has the incorrect sign but is smaller than -1. \( i_c \) denotes the correct class number for the current training pattern, \( i_d \) denotes the incorrect class number for the pattern. \( M \) is the number of class. Then the error function \( E' \) can be presented as

\[ E' = \sum_{p=1}^{N_v} \sum_{i=1}^{M} \left[ t'_p(i) - y_p(i) \right]^2 \quad (5.12) \]
For the correct class,

\[ \text{if } y_p(i_c) \geq t_p(i_c), \text{ then } t_p(i_c) = y_p(i_c) \quad (5.13) \]

For the incorrect class,

\[ \text{if } t_p(i_d) \geq y_p(i_d), \text{ then } t_p(i_d) = y_p(i_d) \quad (5.14) \]

Finally, \( t_p(i) \) is replaced by \( t'_p(i) \), and the error function used in optimizations becomes \( E' \).

5.5 MKM Training Algorithm

The training algorithm consists of the following steps:

Given training data, validation data, and testing data.

1. Initialize RBF neural network, get \( N_h, \beta_k, m_k, R, C, W \).

2. Use validation data to do pruning, then we get updated \( m_k, \beta_k, N_h, W \).

3. Calculate \( Pe_1 \) which is the misclassification error probability of the validation data using the initialized \( \beta_1 \) which is the same as \( \beta_k \) after pruning.

4. Optimize \( \beta_k \) using Newton’s method and output reset method, obtain optimized beta as \( \beta_2 \).

Calculate \( Pe_2 \) which is the misclassification of validation data using the \( \beta_2 \).

5. Compare \( Pe_1 \) and \( Pe_2 \), select \( \beta_1 \) or \( \beta_2 \) which has the minimum validation error.

6. Find \( \lambda \) for of regularization by using validation data.

7. Put training data and validation data together to from a new training data. Use \( \lambda \) and OR to do output weights optimization.

8. Calculate the actual outputs.
Chapter 6

Simulation Results

In this chapter, the results of the MKM algorithm, SVM, LS-SVM are compared. All the simulations shown in this chapter are run in Matlab 2014.

The binary datasets we used can be classified in two groups. One is Diabetes that has 8 inputs and 768 patterns. On the other hand Heart which has 13 inputs which has higher input dimension but 270 patterns which has smaller data size. Next we investigate the performance on multiclass datasets, Iris which has 150 patterns, 2 inputs and 3 classes is small size and low input dimension. Image segment which has 2310 patterns, 19 inputs and 7 classes has large data size and high dimension of inputs. Grng which has 800 patterns, 16 inputs and 4 classes has medium data size but high input dimensions. Ecoli which has 336 patterns, 7 inputs and 8 classes has small data size and medium input dimensions. Vowel which has 990 patterns, 10 inputs but 11 classes has medium data size and input dimension but having the most classes. Table 6.1 summarizes the specifications of the datasets in detail. Table 6.2 shows the testing result after implement each technique. Table 6.3 and 6.4 summarize the k-fold testing performance of the proposed algorithm with other comparable algorithms.
Table 6-1 Specification of datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data size</th>
<th>$N_c$</th>
<th>$N$</th>
<th>$N_V$ for training</th>
<th>$N_V$ for validation</th>
<th>$N_V$ for testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>2</td>
<td>8</td>
<td>691</td>
<td>77</td>
<td>138</td>
</tr>
<tr>
<td>Heart</td>
<td>270</td>
<td>2</td>
<td>13</td>
<td>243</td>
<td>27</td>
<td>49</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>3</td>
<td>4</td>
<td>135</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>Grng</td>
<td>800</td>
<td>4</td>
<td>16</td>
<td>720</td>
<td>80</td>
<td>216</td>
</tr>
<tr>
<td>IS</td>
<td>2310</td>
<td>7</td>
<td>19</td>
<td>2079</td>
<td>231</td>
<td>416</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>8</td>
<td>7</td>
<td>302</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td>Vowel</td>
<td>990</td>
<td>11</td>
<td>10</td>
<td>891</td>
<td>99</td>
<td>178</td>
</tr>
</tbody>
</table>

Table 6-2 Testing result after implement each techniques

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Initialization Success Rate</th>
<th>Pruning Success Rate</th>
<th>Optimize$\beta$ Success Rate</th>
<th>Regularization Success Rate</th>
<th>Output reset Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>54.95</td>
<td>73.95</td>
<td>73.95</td>
<td>73.95</td>
<td>74.86</td>
</tr>
<tr>
<td>Heart</td>
<td>48.15</td>
<td>74.45</td>
<td>74.45</td>
<td>74.45</td>
<td>74.45</td>
</tr>
<tr>
<td>Iris</td>
<td>61.33</td>
<td>96.67</td>
<td>96.67</td>
<td>96.67</td>
<td>98</td>
</tr>
<tr>
<td>Grng</td>
<td>28.375</td>
<td>96.375</td>
<td>96.375</td>
<td>96.5</td>
<td>96.625</td>
</tr>
<tr>
<td>IS</td>
<td>16.80</td>
<td>96.88</td>
<td>96.88</td>
<td>96.88</td>
<td>97.23</td>
</tr>
<tr>
<td>Ecoli</td>
<td>26.67</td>
<td>80.18</td>
<td>80.18</td>
<td>80.76</td>
<td>83.12</td>
</tr>
<tr>
<td>Vowel</td>
<td>99.09</td>
<td>99.09</td>
<td>99.09</td>
<td>99.09</td>
<td>99.19</td>
</tr>
</tbody>
</table>
Table 6- 3 Testing Performance comparison of MKM, LS-SVM, and SVM

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MKM Success Rate (%)</th>
<th>LS-SVM Success Rate (%)</th>
<th>SVM Success Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>74.86</td>
<td>65.11</td>
<td>65.11</td>
</tr>
<tr>
<td>Heart</td>
<td>74.45</td>
<td>55.56</td>
<td>55.66</td>
</tr>
<tr>
<td>Iris</td>
<td>98</td>
<td>94.67</td>
<td>97.33</td>
</tr>
<tr>
<td>Grng</td>
<td>96.625</td>
<td>95.875</td>
<td>95.88</td>
</tr>
<tr>
<td>IS</td>
<td>97.23</td>
<td>95.671</td>
<td>61.08</td>
</tr>
<tr>
<td>Ecoli</td>
<td>83.12</td>
<td>80.81</td>
<td>75.98</td>
</tr>
<tr>
<td>Vowel</td>
<td>99.19</td>
<td>97.68</td>
<td>89.29</td>
</tr>
</tbody>
</table>

Table 6- 4 Number of hidden units for MKM comparison of dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MKM Initial (N_h)</th>
<th>MKM Final (N_h)</th>
<th>LSSVM (N_{LS-SVM})</th>
<th>SVM (N_{SVM})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>553</td>
<td>27.4</td>
<td>691</td>
<td>691</td>
</tr>
<tr>
<td>Heart</td>
<td>194</td>
<td>36.6</td>
<td>243</td>
<td>243</td>
</tr>
<tr>
<td>Iris</td>
<td>122</td>
<td>5.1</td>
<td>135</td>
<td>41.8</td>
</tr>
<tr>
<td>Grng</td>
<td>504</td>
<td>219</td>
<td>720</td>
<td>350.7</td>
</tr>
<tr>
<td>Image Segmentation</td>
<td>1663</td>
<td>256.4</td>
<td>2079</td>
<td>1987.4</td>
</tr>
<tr>
<td>Ecoli</td>
<td>272</td>
<td>33.3</td>
<td>302</td>
<td>220.1</td>
</tr>
<tr>
<td>Vowel</td>
<td>713</td>
<td>325.9</td>
<td>891</td>
<td>771</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusion and Future Work

In the thesis, RBF neural network basis functions are pruned, spread parameters are optimized by Newton’s method and regularization is used for avoid overfitting. Pruning method for basis functions is an improvement over the existing training algorithms. After pruning, we only keep the useful basis functions, so we get fewer support vectors than SVM and LSSVM training methods. The testing errors on all data we present are smaller than SVM and LSSVM training algorithms. The MKM training algorithm is not only simple but also powerful since it requires fewer numbers of hidden units. We can successfully use it to train small but powerful networks for both the two classes and multi-class cases. Although, the proposed training algorithm performs well on small and medium size of data, it has problems with datasets having thousands of patterns.
Appendix A

Description of Data Sets Used For Training, Validation and Testing
I. Pima Indians Diabetes (8 inputs, 2 classes, 768 patterns)

Source from UCI: http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

This data received on 9 May 1990, the original owners are National Institute of Diabetes and
Digestive and Kidney Diseases.

For each feature : (all numeric-valued)

1. Number of times pregnant

2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test

3. Diastolic blood pressure (mm Hg)

4. Triceps skin fold thickness (mm)

5. 2-Hour serum insulin (mu U/ml)

6. Body mass index (weight in kg/(height in m)^2)

7. Diabetes pedigree function

8. Age (years)

9. Class variable (0 or 1)

Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")

II. Heart (13 inputs, 2 classes, 270 patterns)

Source from UCI: http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

For each feature : (all numeric-valued):

1. age

2. sex

3. chest pain type (4 values)
4. resting blood pressure

5. serum cholesterol in mg/dl

6. fasting blood sugar > 120 mg/dl

7. resting electrocardiographic results  (values 0,1,2)

8. maximum heart rate achieved

9. exercise induced angina

10. old peak = ST depression induced by exercise relative to rest

11. the slope of the peak exercise ST segment

12. number of major vessels (0-3) colored by flourosopy

13. thal: 3 = normal; 6 = fixed defect; 7 = reversible defect

14. Class variable (0 or 1)

Class Distribution: (class value 2 is interpreted as "heart disease")

III. Iris (4 inputs, 3 classes, 150 patterns)

Source from UCI: http://archive.ics.uci.edu/ml/datasets/Iris

(a) Creator: R.A. Fisher

(b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)

(c) Date: July, 1988

This data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

For each feature:(all numeric-valued)

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class variable (1,2,3)

Class Distribution: (class value 1 is interpreted as Iris Setosa, class value 2 is interpreted as Iris Versicolour, class value 3 is interpreted as Iris Virginica)

IV. Grng (16 inputs, 4 classes, 800 patterns)

Source from IPNNL LAB, UT Arlington, TX:

http://www.uta.edu/faculty/manry/new_classification.html

The geometric shape recognition data file consists of four geometric shapes, ellipse, triangle, quadrilateral, and pentagon. Each shape consists of a matrix of size 64*64. For each shape, 200 training patterns were generated using different degrees of deformation. The deformations included rotation, scaling, translation, and oblique distortions. The feature set is ring-wedge energy (RNG), and has 16 features.

V. Ecoli (7 inputs, 8 classes, 336 patterns)

Source from UCI: http://archive.ics.uci.edu/ml/datasets/Ecoli

Creator and Maintainer:

Kenta Nakai

Institute of Molecular and Cellular Biology Osaka, University 1-3 Yamada-oka, Suita 565 Japan

nakai@imcb.osaka-u.ac.jp http://www.imcb.osaka-u.ac.jp/nakai/psort.html
Donor: Paul Horton (paulh@cs.berkeley.edu)

Date: September, 1996

Inputs information:

3. lip: von Heijne's Signal Peptidase II consensus sequence score.
4. chg: Presence of charge on N-terminus of predicted lipoproteins.
5. aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
6. alm1: score of the ALOM membrane spanning region prediction program.
7. alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence.

Class Distribution: class value 1 is interpreted as cytoplasm (cp), class value 2 is interpreted as inner membrane without signal sequence (im), class value 3 is interpreted as inner membrane, cleavable signal sequence (imS), class value 4 is interpreted as inner membrane lipoprotein (imL), class value 5 is interpreted as inner membrane, uncleavable signal sequence (imU), class value 6 is interpreted as outer membrane (om), class value 7 is interpreted as outer membrane lipoprotein (omL), class value 8 is interpreted as periplasm (pp).

VI. Image Segmentation (19 inputs, 7 classes, 2310 patterns)

Source from UCI: http://archive.ics.uci.edu/ml/datasets/Image+Segmentation
The instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel.

**Inputs Information:**

1. **region-centroid-col:** the column of the center pixel of the region.
2. **region-centroid-row:** the row of the center pixel of the region.
3. **region-pixel-count:** the number of pixels in a region = 9.
4. **short-line-density-5:** the results of a line extraction algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.
5. **short-line-density-2:** same as short-line-density-5 but counts lines of high contrast, greater than 5.
6. **vedge-mean:** measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.
7. **vedge-sd:** (see 6)
8. **hedge-mean:** measures the contrast of vertically adjacent pixels. Used for horizontal line detection.
9. **hedge-sd:** (see 8).
10. intensity-mean: the average over the region of \((R + G + B)/3\)

11. rawred-mean: the average over the region of the R value.

12. rawblue-mean: the average over the region of the B value.

13. rawgreen-mean: the average over the region of the G value.

14. exred-mean: measure the excess red: \((2R - (G + B))\)

15. exblue-mean: measure the excess blue: \((2B - (G + R))\)

16. exgreen-mean: measure the excess green: \((2G - (R + B))\)

17. value-mean: 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)

18. saturatoin-mean: (see 17)

19. hue-mean: (see 17)

Class Distribution:

Classes: brick face, sky, foliage, cement, window, path, grass.

Ⅶ. Vowel (10 inputs, 11 classes, 990 patterns)

This data is the speaker independent recognition of the eleven steady state vowels of British English

Source from UCI:

http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+%28Vowel+Recognition+-+Deter
 ding+Data%29

David Deterding (data and non-connectionist analysis)

Mahesan Niranjan (first connectionist analysis)
Tony Robinson (description, program, data, and results)

Maintainer: Scott E. Fahlman, CMU
Appendix B

Schmidt procedure
Schmidt procedure

The Schmidt procedure [27][28][29] maps the inputs into an orthonormal system which speeds up the computation of weights. For un-ordered basis functions $\mathbf{x}$ of dimension $N_u$, the $m^{th}$ orthonormal basis function $x^\prime_m$ is defined as [26][27]:

$$x^\prime_m = \sum_{k=1}^{m} a_{mk}x_k$$

Where $A$ is a lower triangular $N_u$ by $N_u$ orthonormal matrix, and $N_u$ equals to $N_h + 1$.

Initially, $x^\prime_1$ is found as $a_{11}x_1$, where

$$a_{11} = \frac{1}{||x_1||} = \frac{1}{[r(1,1)]^{\frac{1}{2}}}$$

For $2 \leq m \leq N_u$, we first perform

$$c_i = \sum_{k=1}^{1} a_{ik} r(k, m)$$

For $1 \leq i \leq m-1$. Second, we set $b_m = 1$ and get:

$$b_k = -\sum_{i=k}^{m-1} c_i a_{ik}$$

For $1 \leq k \leq m-1$. Finally, we get coefficients $a_{mk}$ for the lower triangular matrix $A$ as:

$$a_{mk} = b_k/ \left[ r(m, m) - \sum_{i=1}^{m-1} c_i^2 \right]^{\frac{1}{2}}$$

Then we can get $x^\prime_m$ by using equation (3.18),

Once we get the orthonormal basis functions, the linear mapping weights in the orthonormal system can be simply found as:

$$w^\prime(i,m) = \sum_{k=1}^{m} a_{mk} c(i,k) \quad 1 \leq i \leq M$$

Finally, we can get original system output weights $W$ from orthonormal output weights $W^\prime$ by:
Ordered Basis Function

The raw basis functions $X(n)$ will be reordered as $X_o(n)$ which denotes the $n^{th}$ most useful basis function. Define $o(n)$ is the ordered function which is the index of the $n^{th}$ most useful basis function. Then $X_o(n) = X(o(n))$ for $1 \leq n \leq N_u$.

For $X_o$, we can have [33]

$$X_o = TX$$

or

$$X_o(n) = \sum_{k=1}^{N_u} t(n,k) \cdot X(k)$$

where $T$ is a matrix which can simply reorder the elements of $X$. We consider $T$ is the identity matrix $I$ with its $k^{th}$ and $m^{th}$ rows switched if $T$ forces $X(k)$ and $X(m)$ to change places. The $m^{th}$ row $n^{th}$ column of $I$ is $\delta(m-n)$. Clearly, $t(n,o(n)) = 1$ and other $t(n,k)$ coefficients on the $n^{th}$ row of $T$ are zeros.

The output vector $y$ satisfies [21][33]:

$$y = WX = W_oX_o = W_o'X'$$

Where the weight matrix for the unordered orthonormal system is

$$W' = C^T A^T$$

For ordered basis functions, we have

$$X_o = TX$$

and
\[ X' = AX_o \]

Then we get,

\[ W = W_o T \]

\[ W_o = W_o' A \]

Next, we get output weight matrix for original system as:

\[ W = W_o' AT \]

For the ordered correlation matrices,

\[ R_o = TRT^T \]

\[ C_o = TC \]

For the ordered basis function of orthonormal system, the weight matrix is:

\[ W_o' = C_o^T A^T \]

Finally we get the output weight matrix for original system as:

\[ W = C_o^T A^T A^T \]

Orthonormal system outputs weight are then generated as

\[ w_o'(i, n) = \sum_{k=1}^{N_u} c(o(k), i) \cdot a(n, k) \]

An element of \( T \) is:

\[ t(n, k) = \delta(k - o(n)) \]

We get

\[ at(k, n) = \sum_{u=1}^{N_u} a(k, u) \cdot \delta(n - o(u)) \]

where \( at(k, n) \) is an element of \( AT \).

Then we map \( W_o' \) back to \( W \), we have
\[ w(i, n) = \sum_{k=1}^{N_u} w_o'(i, k) \cdot at(k, n) \]

Replacing \( at(k, n) \), we get:

\[ w(i, n) = \sum_{k=1}^{N_u} w_o'(i, k) \sum_{u=1}^{N_u} a(k, u) \cdot \delta(n - o(u)) \]

Replacing \( n \) by \( o(n) \), we get:

\[ w(i, o(n)) = \sum_{k=1}^{N_u} w_o'(i, k) \sum_{u=1}^{N_u} a(k, u) \cdot \delta(o(n) - o(u)) \]

Simplify the equation, we get:

\[ w(i, o(n)) = \sum_{k=1}^{N_u} w_o'(i, k) \sum_{u=1}^{N_u} a(k, u) \cdot \delta(n - u) \]

Finally, we get the output weights for original system as:

\[ w(i, o(n)) = \sum_{k=1}^{N_u} w_o'(i, k) \cdot a(k, n) \]
Appendix C

Performance Evaluation of RBF Neural Network
Performance Evaluation of RBF Neural Network

1. Training error: Training error is defined as the average error produced by the network when it is subjected to all the patterns that it was trained on.

2. Validation error: Validation error is the average error produced by the network when it is made to process new data not seen during training.

   The training error is usually smaller than the validation error, since the network is already optimized to reduce the error by the validation data during training.

3. Testing error: Testing error is defined as the average error produced by the trained network testing on the test data.
Appendix D

K-fold Cross Validation
K-fold Cross Validation

We use k-fold validation technique for estimating the performance of RBF based classifier.

Given a single data set, we run a single k-fold validation process as follow:

1. Randomly divide data set into k disjoint subsets of equal size where 1 ≤ k ≤ K.
2. for i=1…k, train and validate the classifier using all data which do not belong to fold i.
3. Test the classifier using the fold i.
4. Calculate $E_k$, the number of patterns in fold i which got wrong classification during training.
   Calculate $E_{vk}$, the number of patterns in fold i which got wrong classification during validation. Calculate $E_{tk}$, the number of patterns in fold i which got wrong classification during testing.
5. Repeat the process from step one to step four choosing another i.

To obtain an satisfied accuracy of the classifier, we repeat the k-fold validation for several times. The average of the k training misclassification $P_{e\text{train}}$, validation misclassification $P_{e\text{validation}}$ and testing misclassification $P_{e\text{test}}$ are:

$$P_{e\text{train}} = \frac{E_{\text{train}}}{\text{number of training patterns}}$$

$$P_{e\text{validation}} = \frac{E_{\text{validation}}}{\text{number of validation patterns}}$$

$$P_{e\text{test}} = \frac{E_{\text{test}}}{\text{number of testing patterns}}$$

where $E_{\text{train}}$, $E_{\text{validation}}$ and $E_{\text{test}}$ are defined as

$$E_{\text{train}} = \frac{1}{K} \sum_{k=1}^{K} E_k$$

$$E_{\text{validation}} = \frac{1}{K} \sum_{k=1}^{K} E_{vk}$$

$$E_{\text{test}} = \frac{1}{K} \sum_{k=1}^{K} E_{tk}$$
References


[22] Chih-Jen Lin,"Support Vector Machines for Data Classification ", National Taiwan University, 2004


support vector machine classifiers: some empirical results." *Proceedings of the Twelfth
Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*. 2000.

[44] Suykens, Johan AK, and Joos Vandewalle. "Multiclass least squares support vector
1999.

113-141.


[47] Hastie, Trevor, Robert Tibshirani, and Andreas Buja. "Flexible discriminant analysis by

[48] Platt, John C., Nello Cristianini, and John Shawe-Taylor. "Large Margin DAGs for


[51] Kreßel, Ulrich H-G. "Pairwise classification and support vector machines." *Advances in


[63] NeuralWare, Inc. *Pushing the Frontiers of Neural Computing*. NeuralWare, Inc, Pittsburgh, PA (1991) (Available from NeuralWare, Inc., Penn Center West, Building IV, Suite 227, Pittsburgh, PA 15276, Tel: 412-787-8222.)


Biographical Information

Yilong Hao was born in China in 1987. He obtained his Bachelor of Science degree in Electronic and Information Engineering from Shenyang University of Chemical Technology, Shenyang in July 2010. He came to the U.S. to pursue the Intensive English Program at the English Language Institute of UT Arlington in August 2011. He enrolled in the UT Arlington graduate school in January 2013 to pursue his Master of Science degree in Electrical Engineering. His current research interests include machine learning and pattern recognition.