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

List of Illustrations

List of Tables

List of Symbols

Symbol	**Definition**
$\mathbf{x}$	Input vector
$\mathbf{x}_{\mathrm{p}}$	pth basis function
$\mathbf{t}$	Desired output vector
$\mathbf{t}_{\mathrm{p}}$	pth desired output vector
$\mathrm{t_p}\,(\mathrm{i})$	ith element of $\mathbf{t_p}$
$\mathbf{y_p}$	actual output vector
$\mathrm{y_p}\,(\mathrm{i})$	ith element of $\mathbf{y_p}$
$\mathbf{i}_{\mathrm{c}}$	correct class vector
$\mathrm{i_c(p)}$	correct class number for pth pattern
P	row number in the data file
$\mathrm{N_v}$	number of rows in data file
N	Number of inputs
M	Number of outputs
N_h	Number of hidden units
$\mathrm{N_c}$	number of classes
W	output weight matrix
C	Constant
$\mathrm{w(i,k)}$	weight from the kth hidden unit to the ith output

$\mathbf{m_k}$	kth center vector
R	auto correlation matrix
C	cross correlation matrix
β_k	spread parameter of the kth hidden unit

List of Acronyms

Acronym	**Definition**
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
OR	Output Reset
SVM	Support Vector Machine
LS-SVM	Least Square Support Vector Machine

Chapter 1

Introduction

1.1 Neural Network

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.

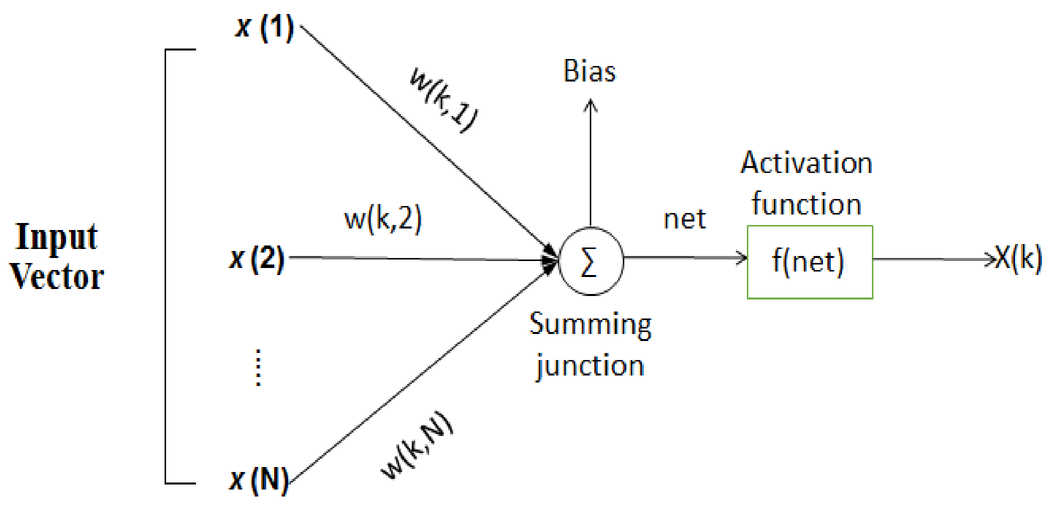

Figure 1-1 A simple nonlinear neuron

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(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.

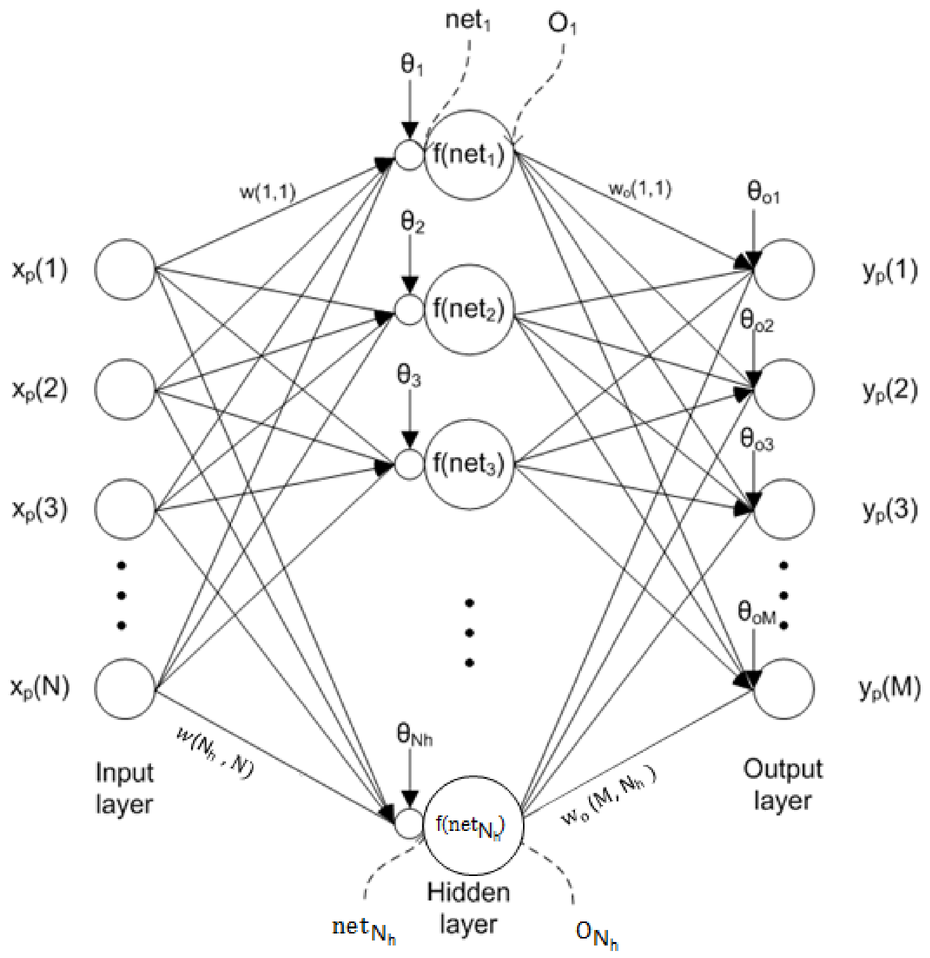

Figure 1-2 Multilayer perceptron with one hidden layer

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 $\mathrm{N_v}$ training patterns $\{\mathbf{x}_\mathrm{p}, \mathrm{i_c(p)}\}$, $1 \leqslant \mathrm{p} \leqslant \mathrm{N_v}$, where $\mathbf{x}_\mathrm{p}$ is the $\mathrm{p^{th}}$ input vector with dimension N, and $\mathrm{i_c(p)}$ is the $\mathrm{p^{th}}$ desired output class label. $\mathrm{i_c(p)}$ is between 1 and M, where M is the total number of classes. The class label $\mathrm{i_c(p)}$ is converted to a desired output vector $\mathbf{t}_\mathrm{p}$ with dimension M as

$$\mathrm{t_p(i)} = \delta\big(\mathrm{i} - \mathrm{i_c(p)}\big) \quad 1 \leq i \leq \mathrm{M} \qquad (2.1)$$

$\mathrm{x_p(n)}$ denotes the $\mathrm{n^{th}}$ element of $\mathbf{x}_\mathrm{p}$. So $\mathbf{x}_\mathrm{p}$ can be represented as $\mathbf{x_p} = \left[\mathrm{x_p(1), x_p(2) \ldots, x_p(N)}\right]^\mathrm{T}$. $\mathrm{t_p(i)}$ denotes the $\mathrm{i^{th}}$ desired output for the $\mathrm{p^{th}}$ input pattern. $\mathbf{y}_\mathrm{p}$ denotes the actual output vector when **x** = $\mathbf{x}_\mathrm{p}$, so $\mathrm{y_p(i)}$ is the $\mathrm{i^{th}}$element of $\mathbf{y}_\mathrm{p}$. $\mathrm{N_h}$ is the number of hidden units. **W** is the output weight matrix and $\mathrm{w(i,k)}$ is the output weight from the $\mathrm{k^{th}}$ hidden unit to the $\mathrm{i^{th}}$ 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.

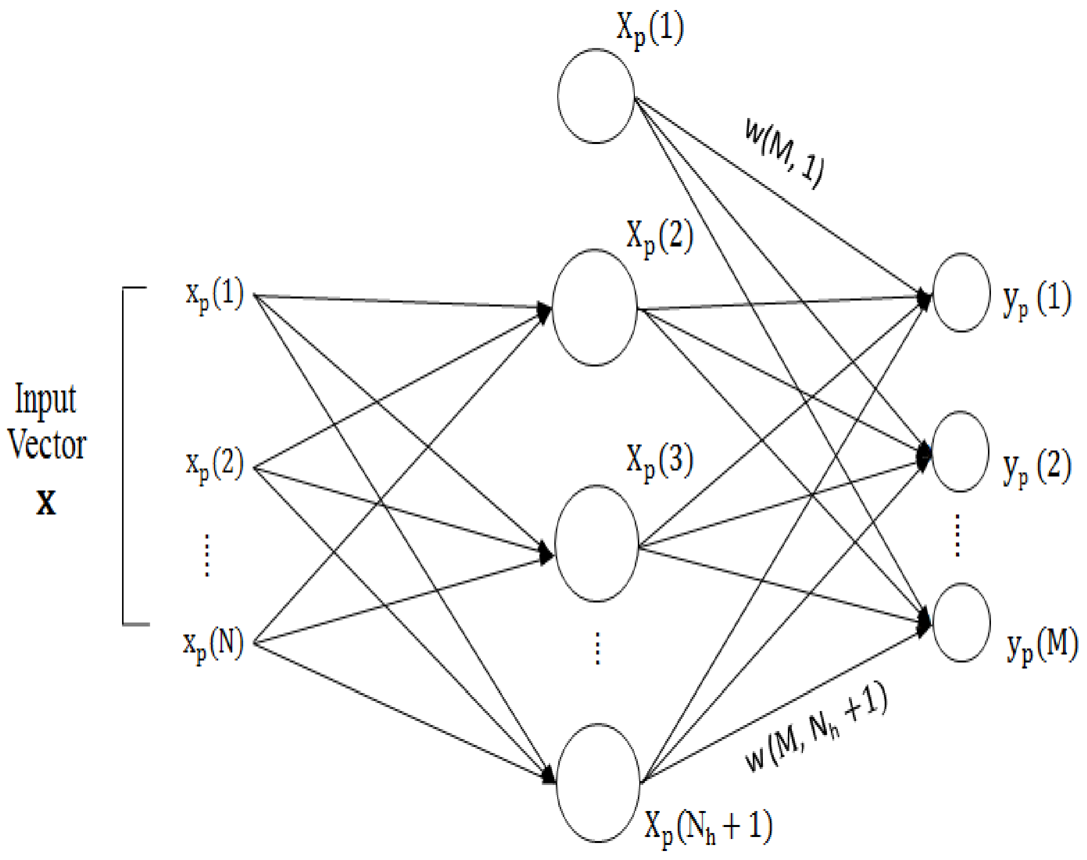

Figure 2-1 Structure of RBF Neural Network with bias

For mapping $\mathbf{x}_p$ to $\mathbf{y}_p$, we have the following steps

(1) Initially we have N_v center vector $\mathbf{m}_k$, which are equal to the input training vectors $\mathbf{x}_k$, where k varies from 1 to N_v. β_k is defined as the spread parameter, μ_k is defined as the mean value of the elements of $\mathbf{m}_k$. For the pth training pattern, $d(\mathbf{x}_p, \mathbf{m}_k)$ is defined as the 2-norm distance between $\mathbf{x}_p$ and $\mathbf{m}_k$[25]:

$$d(\mathbf{x}_p, \mathbf{m}_k) = \sum_{n=1}^{N} (x_p(n) - m_k(n))^2 \qquad (2.2)$$

(2) Initially, the hidden layer consist of the basis vector $\mathbf{X}_p$ of length N_h+1, where $X_p(1) = 1$, and the remaining elements of $\mathbf{X}_p$ are calculated from $\mathbf{x}_p$, the spread parameter β_k, and the center vector $\mathbf{m}_k$ as:

$$X_p(k+1)=\exp\left(-\beta_k d(\mathbf{x}_p, \mathbf{m}_k)\right) \qquad k = 1,2 \ldots N_h \tag{2.3}$$

(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 × (N_h+ 1) weight matrix **W**, and $\mathbf{y}_p$ is calculated as:

$$y_p(i) = \sum_{k=1}^{Nh+1} w(i,k) X_P(k) \tag{2.4}$$

or

$$\mathbf{y}_p = \mathbf{W} \cdot \mathbf{X}_p \tag{2.5}$$

2.3 RBF Neural Network parameter initialization

Given the training data $\{\mathbf{x}_p, \mathbf{t}_p\}$, initialize the center vectors as $\mathbf{m}_k = \mathbf{x}_k$ for k between 1 to N_v. We initialize β_k as[9]

$$\beta_k = 1/(\tfrac{2}{N}\sum_{n=1}^{N}(m_k(n) - \mu_k)^2) \tag{2.6}$$

where

$$\mu_k = \tfrac{1}{N}\sum_{n=1}^{N} m_k(n) \tag{2.7}$$

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

$$E = \frac{1}{N_v}\sum_{p=1}^{N_v} E_p = \frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[t_p(i) - y_p(i)\right]^2 \tag{2.8}$$

2.4 Optimal Output Weights

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

$$r(k,n) = \frac{1}{N_v}\sum_{p=1}^{N_v} X_p(k) \cdot X_p(n) \tag{2.9}$$

The $(\mathrm{N_h}+1) \times \mathrm{M}$ cross-correlation matrix **C** is defined as:

$$c(k,i) = \frac{1}{N_v}\sum_{p=1}^{N_v} X_p(k) \cdot t_p(i) \tag{2.10}$$

The weight matrix **W** can be solved by the following steps:

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

$$E = \frac{1}{N_v}\sum_{p=1}^{N_v} E_p = \frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[t_p(i) - \sum_{k=1}^{N_h+1} w(i,k) \cdot X_p(k)\right]^2 \tag{2.11}$$

Differentiating E with respect to the elements of **W:**

$$\frac{\partial E}{\partial w(m,n)} = \frac{-2}{N_v} \cdot \sum_{p=1}^{N_v}\left(t_p(m) - \sum_{k=1}^{N_h+1} w(m,k) \cdot X_p(k)\right) \cdot X_p(n) \tag{2.12}$$

Then we get:

$$\frac{\partial E}{\partial w(m,n)} = \frac{-2}{N_v} \cdot \sum_{p=1}^{N_v}\left(t_p(m)X_p(n) - \sum_{k=1}^{N_h+1} w(m,k) \cdot X_p(k)X_p(n)\right) \tag{2.13}$$

$$\frac{\partial E}{\partial w(m,n)} = \frac{-2}{N_v} \cdot \left[\sum_{p=1}^{N_v} t_p(m)X_p(n) - \sum_{k=1}^{N_h+1}\left(w(m,k) \cdot \sum_{p=1}^{N_v} X_p(k)X_p(n)\right)\right] \tag{2.14}$$

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

$$\frac{\partial E}{\partial w(m,n)} = \frac{-2}{N_v} \cdot \sum_{p=1}^{N_v}\left(c(n,m) - \sum_{k=1}^{N_h+1} w(m,k) \cdot r(k,n)\right) \tag{2.15}$$

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

$$\mathbf{R} \cdot \mathbf{W}^{\mathrm{T}} = \mathbf{C} \tag{2.16}$$

2.5 Pruning Method

Initially, the center vectors $\mathbf{m}_{\mathrm{k}}$ are initialized as equal to the input vectors as $\mathbf{m}_{\mathrm{k}} = \mathbf{x}_{\mathrm{k}}$, where k varies from 1 to $\mathrm{N_h}$, and $\mathrm{N_h} = \mathrm{N_v}$. Thus, the number of the center vectors are very large since $\mathrm{N_v}$ 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 $\mathbf{X}_{\mathrm{k}}$ will be processed into orthonormal basis function $\mathbf{X'}_{\mathrm{i}}$. $\mathrm{N_u}$ is equal to $\mathrm{N_h} + 1$. For ordered basis function, we get the $\mathrm{m^{th}}$ orthonormal basis function as, [28][30]

$$\mathbf{X}'_m = \sum_{k=1}^{m} a_{mk}\mathbf{X}_{o(k)} \tag{2.17}$$

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

$$a_{11} = \frac{1}{\|\mathbf{X}_{o(1)}\|} = \frac{1}{[r(o(1), o(1))]^{\frac{1}{2}}} \tag{2.18}$$

We get

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

We set $b_m = 1$, then we have

$$b_k = -\sum_{i=k}^{m-1} c_i a_{ik} \quad \text{for } 1 \le i \le m-1 \tag{2.20}$$

$$a_{mk} = b_k \Big/ \left[r\big(o(m), o(m)\big) - \sum_{i=1}^{m-1} {c_i}^2 \right]^{\frac{1}{2}} \quad \text{for } 1 \le k \le m-1 \tag{2.21}$$

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

where $\mathbf{W}_o'$ are the weights for the ordered basis functions. In order to find the most useful basis function, we treat each basis function $\mathbf{X}_k$ 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 $\mathbf{X}_{m_o}$ 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

$$S(m) = \begin{cases} \{\ \emptyset\ \} & \text{for } m = 0 \\ \{o(1), o(2), \ldots, o(m)\} & \text{for } 0 < m \le N_h + 1 \end{cases} \tag{2.23}$$

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:

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

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 :

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

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

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

Then for the general case the candidate basis functions are

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

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 $N_e(m)$ represent the misclassified validation patterns with m hidden units. Let $X_p{}'(m)$ represent the mth ordered orthonormal basis function. For $1 \le i \le M$ and $1 \le m \le N_h + 1$, $y_p(i,m)$ is calculated as

$$y_p(i,1) = w'(i,1) \cdot X_p{}'(1) \tag{2.28}$$

$$y_p(i,m) = y_p(i,m-1) + w'(i,m) \cdot X_p{}'(m) \tag{2.29}$$

Define $i_c'(m) = \operatorname{argmax}_{1 \le i \le M}\left(y_p(i,m)\right)$.

Then $N_e(m)$ is calculated as

$$N_e(m) \leftarrow N_e(m) + \left(1 - \delta\left(i_c' - i_c(p)\right)\right) \tag{2.30}$$

Let P_{ev} represent the misclassification for the validation data with $N_h + 1$ hidden units, where P_{ev} is calculated as:

$$P_{ev}(m) = \frac{N_e(m)}{N_v} \tag{2.31}$$

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 $\mathrm{y_p}$ is calculated as [21]

$$\mathrm{y_p} = \mathbf{w}^{\mathrm{T}} \cdot \mathbf{X}_{\mathrm{p}} - \mathrm{b} \tag{3.1}$$

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

The support vectors are a subset of the basis vectors $\mathbf{X}_{\mathrm{P}}$ for which $\mathrm{y_p} = \mathrm{t_p}$.

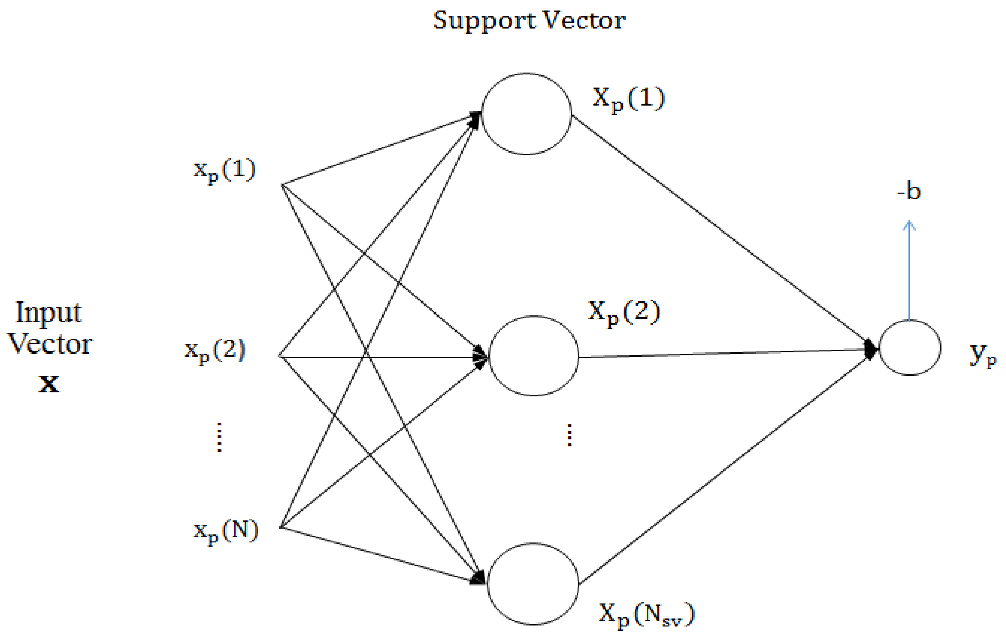

Figure 3-1 Structure of support vector machine

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]:

$$\mathbf{w}^T \cdot \mathbf{X}_p - b = 0 \tag{3.2}$$

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

For a given weight vector **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 $1 \le p \le N_{sv}$

$$\mathbf{w}^T \cdot \mathbf{X}_p - b \ \ge 1 \qquad \text{if } t_p = 1 \tag{3.3}$$

$$\mathbf{w}^{\mathrm{T}} \cdot \mathbf{X}_{\mathrm{p}} - \mathrm{b} \ \leq -1 \qquad \text{if } \mathrm{t}_{\mathrm{p}} = -1 \tag{3.4}$$

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

$$\frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}} \tag{3.5}$$

So, if we want to maximize the margin of separation $2/\|\ \mathbf{w}\ \|$, it is equivalent to minimizing $\|\mathbf{w}\|^2$.

The primal problem is[22]

$$\min E_{svm} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{p=1}^{\mathrm{N_v}} \xi_p \tag{3.6}$$

where C is a user- specified positive parameter, and the ξ_{p} are called the slack variables.We need to find the optimum values of the weight vector **w**, bias b and the slack variables ξ_{p} to minimize $\mathrm{E_{svm}}$. 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 $\{\ \mathbf{x}_{\mathrm{P}}, \mathrm{t}_{\mathrm{p}}\}$, p=1,2, …, $\mathrm{N_v}$, find the Lagrange multipliers α_{p} which maximize the objective function

$$\mathrm{y_p} = \sum_{\mathrm{p=1}}^{\mathrm{N_v}} \alpha_{\mathrm{p}} - \frac{1}{2}\sum_{\mathrm{p=1}}^{\mathrm{N_v}}\sum_{\mathrm{k=1}}^{\mathrm{N_v}} \alpha_{\mathrm{p}}\alpha_{\mathrm{k}}\mathrm{t_p t_k K}(\mathrm{x_p}, \mathrm{x_k}) \tag{3.7}$$

where $\mathrm{K}(\mathrm{x_p}, \mathrm{x_k}) = \mathrm{X_p(k)} = \exp(-\|\mathrm{x_p} - \mathrm{x_k}\|^2/(2\sigma_{\mathrm{k}}{}^2))$

subject to the constraints

$$\sum_{\mathrm{p=1}}^{\mathrm{N_v}} \alpha_{\mathrm{p}}\, \mathrm{y_p} = 0 \tag{3.8}$$

$$0 \leq \alpha_{\mathrm{p}} \leq \mathrm{C} \tag{3.9}$$

where $K(\mathbf{x}_p, \mathbf{x}_k)$ is the RBF kernel function $X_p(k)$ where $\mathbf{m}_k = \mathbf{x}_k$. 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 { $\mathbf{x}_P$,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:

$$\min E_{lssvm} = \frac{1}{2}\|\mathbf{w}\|^2 + \gamma\frac{1}{2}\sum_{p=1}^{N_{sv}}({e_p}^2) \tag{4.1}$$

subject to the equality constraints instead of inequality constraints.

$$y_p\left(\mathbf{w}^T\mathbf{X}_p - b \right) = 1 - e_p \tag{4.2}$$

where γ 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.

$$L(\mathbf{w}, b, \mathbf{e}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{p=1}^{N_v}\alpha_p\left\{y_p\left(\mathbf{w}^T\mathbf{X}_p - b \right) - 1 + e_p\right\} \tag{4.3}$$

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

$$\mathbf{w} = \sum_{p=1}^{N_v}\alpha_p\,\mathbf{X}_p \tag{4.4}$$

$$\sum_{p=1}^{N_v} \alpha_p = 0 \tag{4.5}$$

$$\alpha_p = \gamma e_p \tag{4.6}$$

$$y_p(\mathbf{w}^T \mathbf{X}_p - b) - 1 + e_p = 0 \quad (4.7$$

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 $K(x_p, x_k) = X_p(k) = \exp(-\|x_p - x_k\|^2 / (2\sigma_k{}^2))$

$K(x_p, x_q)$ is the RBF kernel function $X_p(k)$ where $m_k = x_k$.

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,...,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 fucntions. 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 $\mathbf{X}_k{}'$ only keeps the first N_{h_best} units. We then set $N_h = N_{h\_best}$.
3. Find $\mathbf{X}_k$ from the $\mathbf{X}_k{}'$ and **A** matrices as $\mathbf{X}_k = \mathbf{A}^{-1}\mathbf{X}_k{}'$.
4. We rearrange RBF center vectors $\mathbf{m}_k$ and β_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 β_k. In chapter 2 we have initialized the spread parameters and defined them as the inverse of the standard deviation of $\mathbf{m}_k$. We use the error function as in (2.8). For the p^{th} pattern, the k^{th} hidden unit output is:

$$X_p(k+1)=\exp\left(-\beta_k d(\mathbf{x}_p, \mathbf{m}_k)\right) \tag{5.1}$$

The output vector $\mathbf{y}_p$ is the same as described in (2.5).

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

$$g_\beta(k) = \frac{-\partial E}{\partial \beta_k} = \frac{2}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[t_p(i) - y_p(i)\right] \cdot \frac{\partial y_p(i)}{\partial \beta_k} \tag{5.2}$$

where

$$\frac{\partial y_p(i)}{\partial \beta_k} = -\sum_{k=1}^{N_h+1}\sum_{n=1}^{N} w(i,k)\cdot X_p(k)\cdot(x_p(n)-m_k(n))^2 \tag{5.3}$$

Combining (5.29) and (5.30), we get $\mathbf{g}_\beta$. Then we can get the Hessian matrix element as:

$$h_\beta(u,v) = \frac{\partial^2 E}{\partial\beta_u\,\partial\beta_v} = \frac{2}{N_v}\sum_{p=1}^{N_v}\left[\sum_{i=1}^{M}\frac{\partial y_p(i)}{\partial\beta_u}\cdot\frac{\partial y_p(i)}{\partial\beta_v}\right] \tag{5.4}$$

We get the following equation

$$\mathbf{H}_\beta\cdot\mathbf{z} = \mathbf{g}_\beta \tag{5.5}$$

Solving by using OLS, we can get $\mathbf{z}$, β_k can be updated as

$$\beta_k \leftarrow \beta_k + z_k \tag{5.6}$$

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:

$$\begin{aligned} E &= \frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[t_p(i)-y_p(i)\right]^2 + \lambda \parallel \mathbf{w} \parallel^2 \\ &= \frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{M}\left[t_p(i)-\sum_{k=1}^{N_h+1} w(i,k)\cdot X_p(k)\right]^2 + \lambda \parallel \mathbf{w} \parallel^2 \end{aligned} \tag{5.7}$$

Differentiating E with respect to the elements of $\mathbf{w}$, and writing equation in terms of the auto correlation and cross correlation matrices we have:

$$\frac{\partial E}{\partial w(m,n)} = \frac{-2}{N_v}\cdot\sum_{p=1}^{N_v}\left(c(n,m)-\sum_{k=1}^{N_h+1} w(m,k)\cdot r(k,n)\right) + 2\,\lambda\, w(m,n) \tag{5.8}$$

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:

$$\mathbf{C} - \mathbf{R} \cdot \mathbf{W}^{\mathrm{T}} = \lambda \mathbf{W} \tag{5.9}$$

or

$$\mathbf{W} = (\mathbf{R} + \lambda \mathbf{I})^{-\mathbf{1}} \mathbf{C} \tag{5.10}$$

We then minimize E_v with respect to λ, 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.

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

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

$$E' = \sum_{p=1}^{N_v} \sum_{i=1}^{M} \left[t'_p(i) - y_p(i) \right]^2 \tag{5.12}$$

For the correct class,

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

For the incorrect class,

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

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, β_k, $\mathbf{m}_k$, **R, C, W,**

2. Use validation data to do pruning, then we get updated $\mathbf{m}_k$, β_k, N_h, **W**

3. Calculate Pe_1 which is the misclassification error probability of the validation data using the initialized β_1 which is the same as β_k after pruning

4. Optimize β_k using Newton's method and output reset method, obtain optimized beta as β_2. Calculate Pe_2 which is the misclassification of validation data using the β_2.

5. Compare Pe_1 and Pe_2, select β_1 or β_2 which has the minimum validation error.

6. Find λ for of regularization by using validation data

7. Put training data and validation data together to from a new training data. Use λ 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

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

Table 6- 2 Testing result after implement each techniques

Dataset	Initialization Success Rate	Pruning Success Rate	Optimizeβ Success Rate	Regularization Success Rate	Output reset Success Rate
Diabetes	54.95	**73.95**	73.95	73.95	**74.86**
Heart	48.15	**74.45**	74.45	74.45	74.45
Iris	61.33	**96.67**	96.67	96.67	**98**
Grng	28.375	**96.375**	96.375	**96.5**	**96.625**
IS	16.80	**96.88**	96.88	96.88	**97.23**
Ecoli	26.67	**80.18**	80.18	**80.76**	**83.12**
Vowel	99.09	99.09	99.09	99.09	**99.19**

Table 6- 3 Testing Performance comparison of MKM, LS-SVM, and SVM

Dataset	MKM Success Rate (%)	LS-SVM Success Rate (%)	SVM Success Rate (%)
Diabetes	**74.86**	65.11	65.11
Heart	**74.45**	55.56	55.66
Iris	**98**	94.67	97.33
Grng	**96.625**	95.875	95.88
IS	**97.23**	95.671	61.08
Ecoli	**83.12**	80.81	75.98
Vowel	**99.19**	97.68	89.29

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

Dataset	MKM Initial $\mathrm{N_h}$	MKM Final $\mathrm{N_h}$	LSSVM $\mathrm{N_{LS-SVM}}$	SVM $\mathrm{N_{SVM}}$
Diabetes	553	**27.4**	691	691
Heart	194	**36.6**	243	243
Iris	122	**5.1**	135	41.8
Grng	504	**219**	720	350.7
Image Segmentation	1663	**256.4**	2079	1987.4
Ecoli	272	**33.3**	302	220.1
Vowel	713	**325.9**	891	771

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 = reversable 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

Institue 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:

1. mcg: McGeoch's method for signal sequence recognition.
2. gvh: von Heijne's method for signal sequence recognition.
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 perisplasm (pp).

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

Source from UCI: http://archive.ics.uci.edu/ml/datasets/Image+Segmentation

Creators: Vision Group, University of Massachusetts

Donor: Vision Group (Carla Brodley, brodley@cs.umass.edu)

Date: November, 1990

Relevant Information:

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 extractoin 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. vegde-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 **x** of dimension N_u, the mth orthonormal basis function x'_m is defined as [26][27]:

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

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

Initially , x'_1 is found as $a_{11}x_1$, where

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

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

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

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

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

For $1 \le k \le m-1$. 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'_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:

$$w'(i,m) = \sum_{k=1}^{m} a_{mk}\, c(i,k) \qquad 1 \le i \le M$$

Finally, we can get original system output weights **W** from orthonormal output weights $\mathbf{W}'$ by:

$$w(m,k) = \sum_{m=k}^{N_u} a_{mk}\, w'(i,m)$$

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 ≤ n ≤ N_u.

For $\mathbf{X}_o$, we can have [33]

$$\mathbf{X}_o = \mathbf{TX}$$

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 δ(m-n). Clearly, t(n,o(n)) = 1 and other t(n,k) coefficients on the nth row of **T** are zeros.

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

$$\mathbf{y} = \mathbf{WX} = \mathbf{W}_o\mathbf{X}_o = \mathbf{W}_o{}'\mathbf{X}'$$

Where the weight matrix for the unordered orthonormal system is

$$\mathbf{W}' = \mathbf{C}^T\mathbf{A}^T$$

For ordered basis functions, we have

$$\mathbf{X}_o = \mathbf{TX}$$

and

$$\mathbf{X}' = \mathbf{A}\mathbf{X}_\mathrm{o}$$

Then we get,

$$\mathbf{W} = \mathbf{W}_\mathrm{o}\mathbf{T}$$

$$\mathbf{W}_\mathrm{o} = \mathbf{W}_\mathrm{o}'\mathbf{A}$$

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

$$\mathbf{W} = \mathbf{W}_\mathbf{o}'\mathbf{A}\mathbf{T}$$

For the ordered correlation matrices,

$$\mathbf{R}_\mathrm{o} = \mathbf{T}\mathbf{R}\mathbf{T}^\mathbf{T}$$

$$\mathbf{C}_\mathrm{o} = \mathbf{T}\mathbf{C}$$

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

$$\mathbf{W}_\mathrm{o}' = \mathbf{C}_\mathrm{o}{}^\mathrm{T}\mathbf{A}^\mathrm{T}$$

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

$$\mathbf{W} = \mathbf{C}_\mathrm{o}{}^\mathrm{T}\mathbf{A}^\mathrm{T}\mathbf{A}\mathbf{T}$$

Orthonormal system outputs weight are then generated as

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

An element of **T** is:

$$\mathrm{t}(\mathrm{n},\mathrm{k}) = \delta\big(\mathrm{k} - \mathrm{o}(\mathrm{n})\big)$$

We get

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

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

Then we map $\mathbf{W}_\mathrm{o}'$ 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

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

Replacing n by o(n), we get

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

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 $1 \leq k \leq K$.
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 $\mathrm{E_k}$, the number of patterns in fold i which got wrong classification during training. Calculate $\mathrm{E_{vk}}$, the number of patterns in fold i which got wrong classification during validation. Calculate $\mathrm{E_{tk}}$, 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 $\mathrm{Pe_{train}}$, validation misclassification $\mathrm{Pe_{validation}}$ and testing misclassification $\mathrm{Pe_{test}}$ are:

$$\mathrm{Pe_{train}} = \frac{\mathrm{E_{train}}}{\text{number of training patterns}}$$

$$\mathrm{Pe_{validation}} = \frac{\mathrm{E_{validation}}}{\text{number of validation patterns}}$$

$$\mathrm{Pe_{test}} = \frac{\mathrm{E_{test}}}{\text{number of testing patterns}}$$

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

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

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

$$\mathrm{E_{test}} = \tfrac{1}{\mathrm{K}}\textstyle\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{E_{tk}}$$

References

[1] M. Egmont-Petersena , D. de Ridderb and H. Handelsc, “Image processing with neural networks—a review”, The Journal of The Pattern Recognition Society, 21 August 2011.

[2] Derrick H.Nguyen and Bernard Widrow, “Neural Networks for Self-Learning Control Systems”, IEEE Control Systems Magazine, April 1990.

[3] J.Jiang, P.Trundle and J.Ren, “Medical image analysis with artificial neural networks” , journal of Computerized Medical Imaging and Graphics, July 2010.

[4] Soumitro Swapan Auddy, “Discriminant Processing in License Plate Recognition”, Thesis presented at UT Arlington, December 2013.

[5] Eldon.Y.Li,” Artificial neural networks and their business applications”, journal of Information & Management, 1994.

[6] M. W. Craven and J. W. Shavlik. (1997, Using neural networks for data mining. FGCS.Future Generations Computer Systems 13(2-3), pp. 211-229.

[7] H. Lu, R. Setiono and H. Liu. (1996, Effective data mining using neural networks. IEEE Trans. Knowled. Data Eng. 8(6), pp. 957-961.

[8] S. Lawrence, C. L. Giles, A. C. Tsoi and A. D. Back. (1997, Face recognition: A convolutional neural-network approach. IEEE Trans. Neural Networks 8(1), pp. 98-113.

[9] S. Haykin. (Neural Networks and learning machines, third edition).

[10] Rohit Rawat, “An Effective Piecewise Linear Network,” Thesis presented at UT Arlington, pp. 1-8, December 2009.

[11] F. L. Lewis, S. Jagannathan and A. Yeşildirek. (1998, Neural Network Control of Robot Manipulators and Nonlinear Systems.

[12] S. Chen and S. A. Billings. Neural networks for non-linear dynamic system modelling and identification. Advances in Intelligent Control

[13] T. Y. Kwok and D. Y. Yeung. (1997, Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Trans. Neural Networks

[14] C. H. Sequin and R. D. Clay. Fault tolerance in artificial neural networks. Presented at Neural Networks, 1990., 1990 IJCNN International Joint Conference on.

[15] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST) 2.3 (2011): 27.

[16] Ye, Jieping, and Tao Xiong. "SVM versus least squares SVM." International Conference on Artificial Intelligence and Statistics. 2007.

[17] Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition." Data mining and knowledge discovery 2.2 (1998): 121-167.

[18] Christiannini, N., and J. Shawe-Taylor. "Support vector machines and other kernel-based learning methods." (2000).

[19] Schoslkopf, Bernhard, and Alexander Smola. "Learning with Kernels, Support Vector Machines." (2002).

[20] Vapnik, Vladimir. The nature of statistical learning theory. Springer Science & Business Media, 2000.

[21] Michael T. Manry, “Pattern Recognition II”, IPNNL UT Arlington, 2013

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

[23] Vapnik, Vladimir Naumovich, and Vlamimir Vapnik. Statistical learning theory. Vol. 1. New York: Wiley, 1998.

[24] Fletcher, Roger. *Practical methods of optimization*. John Wiley & Sons, 2013

[25] S. Haykin, Neural Networks: A Comprehensive Foundation, by, Macmillan Publishing Company, Inc.,1994

[26] J. W. Dettman. (1988, *Mathematical Methods in Physics and Engineering.*

[27] Maldonado, F. J., and M. T. Manry. "Optimal pruning of feedforward neural networks based upon the Schmidt procedure." *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*. Vol. 2. IEEE, 2002.

[28] Chen, Sheng, Colin FN Cowan, and Peter M. Grant. "Orthogonal least squares learning algorithm for radial basis function networks." *Neural Networks, IEEE Transactions on* 2.2 (1991): 302-309.

[29] Chen, Sheng, Stephen A. Billings, and Wan Luo. "Orthogonal least squares methods and their application to non-linear system identification." *International Journal of control* 50.5 (1989): 1873-1896.

[30] Rawat, Rohit, Jignesh K. Patel, and Michael T. Manry. "Minimizing validation error with respect to network size and number of training epochs." *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013.

[31] Maldonado, F. J., M. T. Manry, and Tae-Hoon Kim. "Finding optimal neural network basis function subsets using the Schmidt procedure." *Neural Networks, 2003. Proceedings of the International Joint Conference on*. Vol. 1. IEEE, 2003.

[32] Narasimha, Pramod L., et al. "An integrated growing-pruning method for feedforward network training." *Neurocomputing* 71.13 (2008): 2831-2847.

[33] Michael T. Manry, “OLS for Ordered Basis Functions”, IPNNL UT Arlington, 2014

[34] Narasimha, Pramod Lakshmi, et al. "Fast Generation of a Sequence of Trained and Validated Feed-Forward Networks." *FLAIRS Conference*. 2006.

[35] Orr, Mark JL. "Regularization in the selection of radial basis function centers."*Neural computation* 7.3 (1995): 606-623.

[36] Schölkopf, Bernhard, and Christopher JC Burges. *Advances in kernel methods: support vector learning*. MIT press, 1999.

[37] Vapnik, Vladimir. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[38] Vapnik, Vladimir Naumovich, and Vlamimir Vapnik. *Statistical learning theory*. Vol. 1. New York: Wiley, 1998.

[39] Vapnik, Vladimir. "The support vector method of function estimation." *Nonlinear Modeling*. Springer US, 1998. 55-85.

[40] Van Gestel, Tony, et al. "Benchmarking least squares support vector machine classifiers." *Machine Learning* 54.1 (2004): 5-32.

[41] Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." *Neural processing letters* 9.3 (1999): 293-300.

[42] Ye, Jieping, and Tao Xiong. "SVM versus least squares SVM." *International Conference on Artificial Intelligence and Statistics*. 2007.

[43] Baesens, B., et al. "An initial approach to wrapped input selection using least squares 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 machines." *Neural Networks, 1999. IJCNN'99. International Joint Conference on*. Vol. 2. IEEE, 1999.

[45] Allwein, Erin L., Robert E. Schapire, and Yoram Singer. "Reducing multiclass to binary: A unifying approach for margin classifiers." *The Journal of Machine Learning Research* 1 (2001): 113-141.

[46] Dietterich, Thomas G., and Ghulum Bakiri. "Solving multiclass learning problems via error-correcting output codes." *Journal of artificial intelligence research* (1995): 263-286.

[47] Hastie, Trevor, Robert Tibshirani, and Andreas Buja. "Flexible discriminant analysis by optimal scoring." *Journal of the American statistical association*89.428 (1994): 1255-1270.

[48] Platt, John C., Nello Cristianini, and John Shawe-Taylor. "Large Margin DAGs for Multiclass Classification." *nips*. Vol. 12. 1999.

[49] Sejnowski, Terrence J., and Charles R. Rosenberg. "Parallel networks that learn to pronounce English text." *Complex systems* 1.1 (1987): 145-168.

[50] Utschick, Wolfgang. "A regularization method for non-trivial codes in polychotomous classification." *International Journal of Pattern Recognition and Artificial Intelligence* 12.04 (1998): 453-474.

[51] Kreßel, Ulrich H-G. "Pairwise classification and support vector machines."*Advances in kernel methods*. MIT Press, 1999.

[52] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

[53] Colmenarez, Antonio, Brendan Frey, and Thomas S. Huang. "Detection and tracking of faces and facial features." *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*. Vol. 1. IEEE, 1999.

[54] Maglaveras, Nicos, et al. "ECG pattern recognition and classification using non-linear transformations and neural networks: a review." *International journal of medical informatics* 52.1 (1998): 191-208.

[55] Wang, Haifeng, and Dejin Hu. "Comparison of SVM and LS-SVM for regression." *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*. Vol. 1. IEEE, 2005.

[56] Mathur, A., and G. M. Foody. "Multiclass and binary SVM classification: Implications for training and classification users." *Geoscience and Remote Sensing Letters, IEEE* 5.2 (2008): 241-245.

[57] Li, Jiang, Michael T. Manry, Li-Min Liu, Changhua Yu, and John Wei. "Iterative Improvement of Neural Classifiers." *FLAIRS Conference*. 2004.

[58] Odom, Marcus D., and Ramesh Sharda. "A neural network model for bankruptcy prediction." *1990 IJCNN International Joint Conference on neural networks*. 1990.

[59] Koster, A., N. E. Sondak, and W. Bourbia. "A business application of artificial neural network systems." *J. COMP. INF. SYST.* 31.2 (1991): 3-9.

[60] Salchenberger, Linda M., E. Cinar, and Nicholas A. Lash. "Neural networks: A new tool for predicting thrift failures*." *Decision Sciences* 23.4 (1992): 899-916.

[61] Chen, An-Sing, Mark T. Leung, and Hazem Daouk. "Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index." *Computers & Operations Research* 30.6 (2003): 901-923.

[62] Lawrence, Ramon. "Using neural networks to forecast stock market prices."*University of Manitoba* (1997).

[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.)

[64] Guardado, J. L., et al. "A comparative study of neural network efficiency in power transformers diagnosis using dissolved gas analysis." *Power Delivery, IEEE Transactions on* 16.4 (2001): 643-647.

[65] Khan, Javed, et al. "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks." *Nature medicine* 7.6 (2001): 673-679.

[66] De Brabanter, Kris. "Least squares support vector regression with applications to large-scale data: a statistical approach." *Faculty of Engineering, KU Leuven, Katholieke Universiteit Leuven* (2011).

[67] Parisi, R., et al. "Car plate recognition by neural networks and image processing." *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*. Vol. 3. IEEE, 1998.

[68] Koval, V., et al. "Smart license plate recognition system based on image processing using neural network." Intelligent Data Acquisition and Advanced Computing Systems: Technology

and Applications, 2003. Proceedings of the Second IEEE International Workshop on. IEEE, 2003.

[69] Zhou, Yi-Tong, et al. "Image restoration using a neural network." *Acoustics, Speech and Signal Processing, IEEE Transactions on* 36.7 (1988): 1141-1151.

[70] Daugman, John G. "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression." *Acoustics, Speech and Signal Processing, IEEE Transactions on* 36.7 (1988): 1169-1179.

[71] Kulkarni, Arun D. *Artificial neural networks for image understanding*. John Wiley & Sons, Inc., 1993.

[72] Lawrence, Steve, et al. "Face recognition: A convolutional neural-network approach." *Neural Networks, IEEE Transactions on* 8.1 (1997): 98-113.

[73] Carpenter, Gail A. "Neural network models for pattern recognition and associative memory." *Neural networks* 2.4 (1989): 243-257.

[74] Zhang, G. Peter. "Time series forecasting using a hybrid ARIMA and neural network model." *Neurocomputing* 50 (2003): 159-175.

[75] Dony, Robert D., and Simon Haykin. "Neural network approaches to image compression." *Proceedings of the IEEE* 83.2 (1995): 288-303.

[76] Lin, Chin-Teng, and CS George Lee. "Neural-network-based fuzzy logic control and decision system." *Computers, IEEE Transactions on* 40.12 (1991): 1320-1336.

[77] Hunt, K. Jetal, et al. "Neural networks for control systems—a survey."*Automatica* 28.6 (1992): 1083-1112.

[78] Tanaka, Kazuo. "An approach to stability criteria of neural-network control systems." *Neural Networks, IEEE Transactions on* 7.3 (1996): 629-642.

[79] Narendra, Kumpati S., and Kannan Parthasarathy. "Identification and control of dynamical systems using neural networks." *Neural Networks, IEEE Transactions on* 1.1 (1990): 4-27.

[80] Nguyen, Derrick H., and Bernard Widrow. "Neural networks for self-learning control systems." *Control Systems Magazine, IEEE* 10.3 (1990): 18-23.

[81] Miller, W. Thomas, Paul J. Werbos, and Richard S. Sutton. *Neural networks for control*. MIT press, 1995.

[82] Hall, Lawrence O., et al. "A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain." *Neural Networks, IEEE Transactions on* 3.5 (1992): 672-682.

[83] Özkan, Mehmed, Benoit M. Dawant, and Robert J. Maciunas. "Neural-network-based segmentation of multi-modal medical images: a comparative and prospective study." *Medical Imaging, IEEE Transactions on* 12.3 (1993): 534-544.

[84] Reddick, Wilburn E., et al. "Automated segmentation and classification of multispectral magnetic resonance images of brain using artificial neural networks." *Medical Imaging, IEEE Transactions on* 16.6 (1997): 911-918.

[85] Ahmed, Mohamed N., and Aly A. Farag. "Two-stage neural network for volume segmentation of medical images." *Pattern Recognition Letters* 18.11 (1997): 1143-1151.

[86] Smith, Brian L., and Michael J. Demetsky. "Short-term traffic flow prediction: neural network approach." *Transportation Research Record* 1453 (1994).

[87] Kneller, D. G., F. E. Cohen, and R. Langridge. "Improvements in protein secondary structure prediction by an enhanced neural network." *Journal of molecular biology* 214.1 (1990): 171-182.

[88] Schaap, Marcel G., Feike J. Leij, and Martinus Th van Genuchten. "Neural network analysis for hierarchical prediction of soil hydraulic properties." *Soil Science Society of America Journal* 62.4 (1998): 847-855.

[89] Martin, Gale L. "Pattern recognition neural network." U.S. Patent No. 5,440,651. 8 Aug. 1995.

[90] Bishop, Christopher M. *Neural networks for pattern recognition*. Oxford university press, 1995.

[91] Looney, Carl Grant. *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*. Oxford University Press, Inc., 1997.

[92] Kamijo, Ken-ichi, and Tetsuji Tanigawa. "Stock price pattern recognition-a recurrent neural network approach." *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*. IEEE, 1990.

[93] Rowley, Henry, Shumeet Baluja, and Takeo Kanade. "Neural network-based face detection." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20.1 (1998): 23-38.

[94] Waibel, Alexander, et al. "Phoneme recognition using time-delay neural networks." *Acoustics, Speech and Signal Processing, IEEE Transactions on*37.3 (1989): 328-339.

[95] Lin, Shang-Hung, Sun-Yuan Kung, and Long-Ji Lin. "Face recognition/detection by probabilistic decision-based neural network." *Neural Networks, IEEE Transactions on* 8.1 (1997): 114-132.

[96] Jagielska, Ilona, and Janusz Jaworski. "Neural network for predicting the performance of credit card accounts." *Computational Economics* 9.1 (1996): 77-82.

[97] Baesens, Bart, et al. "Using neural network rule extraction and decision tables for credit-risk evaluation." *Management science* 49.3 (2003): 312-329.

[98] Lee, Kun Chang, Ingoo Han, and Youngsig Kwon. "Hybrid neural network models for bankruptcy predictions." *Decision Support Systems* 18.1 (1996): 63-72.

[99] Wilson, Rick L., and Ramesh Sharda. "Bankruptcy prediction using neural networks." *Decision support systems* 11.5 (1994): 545-557.

[100] Yoon, Youngohc, and George Swales. "Predicting stock price performance: A neural network approach." *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*. Vol. 4. IEEE, 1991.

[101] Kara, Yakup, Melek Acar Boyacioglu, and Ömer Kaan Baykan. "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange." *Expert systems with Applications* 38.5 (2011): 5311-5319.

[102] Dutta, Goutam, et al. "Artificial neural network models for forecasting stock price index in the Bombay stock exchange." *Journal of Emerging Market Finance* 5.3 (2006): 283-295.

[103] Zekic, Marijana. "Neural network applications in stock market predictions-a methodology analysis." *proceedings of the 9th International Conference on Information and Intelligent Systems*. Vol. 98. 1998.

[104] Khoa, Nguyen Lu Dang, Kazutoshi Sakakibara, and Ikuko Nishikawa. "Stock price forecasting using back propagation neural networks with time and profit based adjusted weight factors." *SICE-ICASE, 2006. International Joint Conference*. IEEE, 2006.

[105] Dase, R. K., and D. D. Pawar. "Application of Artificial Neural Network for stock market predictions: A review of literature." *International Journal of Machine Intelligence* 2.2 (2010): 14-17.

[106] Gharbi, R. B., and Adel M. Elsharkawy. "Neural network model for estimating the PVT properties of Middle East crude oils." *Middle East Oil Show and Conference*. Society of Petroleum Engineers, 1997.

[107] Osman, El-Sayed A., and Muhammad Ali Al-Marhoun. "Artificial neural networks models for predicting PVT properties of oil field brines." *SPE Middle East Oil and Gas Show and Conference*. Society of Petroleum Engineers, 2005.

[108] Moghadam, J. Naseryan, K. Salahshoor, and R. Kharrat. "Introducing a new method for predicting PVT properties of Iranian crude oils by applying artificial neural networks." *Petroleum Science and Technology* 29.10 (2011): 1066-1079.

[109] Elsharkawy, Adel M. "Modeling the properties of crude oil and gas systems using RBF network." *SPE Asia Pacific Oil and Gas Conference and Exhibition*. Society of Petroleum Engineers, 1998.

[110] Karabatak, Murat, and M. Cevdet Ince. "An expert system for detection of breast cancer based on association rules and neural network." *Expert Systems with Applications* 36.2 (2009): 3465-3469.

[111] Burke, Harry B., et al. "Artificial neural networks improve the accuracy of cancer survival prediction." *Cancer* 79.4 (1997): 857-862.

[112] Baker, Jay A., et al. "Breast cancer: prediction with artificial neural network based on BI-RADS standardized lexicon." *Radiology* 196.3 (1995): 817-822.

[113] Lo, Joseph Y., et al. "Predicting breast cancer invasion with artificial neural networks on the basis of mammographic features." *Radiology* 203.1 (1997): 159-163.

[114] Bottaci, Leonardo, et al. "Artificial neural networks applied to outcome prediction for colorectal cancer patients in separate institutions." *The Lancet*350.9076 (1997): 469-472.

[115] Ahmed, Farid E. "Artificial neural networks for diagnosis and survival prediction in colon cancer." *Molecular cancer* 4.1 (2005): 29.

[116] Sug, Hyontai. "Performance Comparison of RBF networks and MLPs for Classification." *Proceedings of the 9th WSEAS International Conference on applied Informatics and Communications (AIC'09)*. 2009.

[117] Mayoraz, Eddy, and Ethem Alpaydin. "Support vector machines for multi-class classification." *Engineering Applications of Bio-Inspired Artificial Neural Networks*. Springer Berlin Heidelberg, 1999. 833-842.

[118] Luts, Jan, et al. "A tutorial on support vector machine-based methods for classification problems in chemometrics." *Analytica Chimica Acta* 665.2 (2010): 129-145.

[119] Baudat, Gaston, and Fatiha Anouar. "Kernel-based methods and function approximation." *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*. Vol. 2. IEEE, 2001.

[120] Shawe-Taylor, John, and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[121] Kuh, Anthony. "Least squares kernel methods and applications." *Soft Computing in Communications*. Springer Berlin Heidelberg, 2004. 365-387.

[122] Smola, Alex J., and Bernhard Schölkopf. "A tutorial on support vector regression." *Statistics and computing* 14.3 (2004): 199-222.

[123] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, 2002 (ISBN 981-238-151-1)

[124] Suykens, Johan AK, et al. "Weighted least squares support vector machines: robustness and sparse approximation." *Neurocomputing* 48.1 (2002): 85-105.

[125] Gore, R. G., et al. "Iterative design of neural network classifiers through regression." *International Journal on Artificial Intelligence Tools* 14.01n02 (2005): 281-301.

[126] Orr, Mark JL. "Regularization in the selection of radial basis function centers."*Neural computation* 7.3 (1995): 606-623.

[127] Tyagi, Kanishka. "Second Order Training Algorithms For Radial Basis Function Neural Networks." (2012).

[128] Cristianini, Nello, and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

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.