SUPPORT VECTOR MACHINE IN CHAOTIC HYDROLOGICAL TIME SERIES FORECASTING

YU XINYING

NATIONAL UNIVERSITY OF SINGAPORE

2004

SUPPORT VECTOR MACHINE IN CHAOTIC HYDROLOGICAL TIME SERIES FORECASTING

YU XINYING

(M. SC., UNESCO-IHE, DISTINCTION)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF CIVIL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

ACKNOWLEDGEMENTS

I wish to express my sincerer and deep gratitude to my supervisor, Assoc. Prof. Liong Shie-Yui, for his inspiration and supervision during my PhD study at The National University of Singapore. Uncounted number of discussions leads to the various techniques shown in this thesis. His invaluable advices, suggestions, guidance and encouragement are highly appreciated. His great supervisions undoubtedly make my PhD study fruitful and an enjoyable experience.

I am grateful to my co-supervisor, Dr. Vladan Babovic, for sharing his ideas throughout the study period.

I also wish to thank Assoc. Prof. Phoon Kok Kwang for his concerns, comments and discussions.

I am grateful to Prof. M. B. Abbott for his genuine concerns on my study and well-being during this study period.

I would like to thank the examiners for their valuable corrections, suggestions, and comments.

Thanks are extended to Assoc. Prof. S. Sathiya Keerthi for his great Neural Networks course. Many thanks also to laboratory technician of Hydraulics Lab, Mr. Krishna, for his assistance.

I would also like to thank my friends together with whom I had a wonderful time in Singapore. They are: Hu Guiping, Yang Shufang and Zhao Ying. Thanks are also extended to Lin Xiaohan, Zhang Xiaoli, Li Ying, Chen Jian, Ma Peifeng, He Jiangcheng, Doan Chi Dung, Dulakshi Karunasingha, Anuja, Sivapragasam, and all colleagues in Hydraulic Lab in NUS. In addition, I am grateful to Xu Min, Qin Zhen and Nguyen Huu Hai for their valuable suggestions on some implementation of techniques in C or FORTRAN under Windows.

Heartfelt thanks to my dear parents and my family in China, who continuously support me with their love. Special thanks to my friends He Hai, Zhao Hongli, Wang Ping, You Aiju for their forever friendship.

I would like to thank to all persons who have contributed to the success of this study. Finally I would like to acknowledge my appreciation to National University of Singapore for the financial support received through the NUS research scholarship. In addition, the great library and digital library facilities deserve some special mention.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

TABLE OF CONTENTS iii

SUMMARY vii

NOMENCLATURE ix

LIST OF FIGURES xii

LIST OF TABLES xv

CHAPTER 1 INTRODUCTION 1

1.1 Background 1

1.2 Need for the present study 3

1.2.1 Support vector machine for phase space reconstruction 4

1.2.2 Handling large chaotic data sets efficiently 5

1.2.3 Automatic parameter calibration 6

1.3 Objectives of the present study 7

1.4 Thesis organization 8

CHAPTER 2 LITERATURE REVIEW 10

2.1 Introduction 10

2.2 Chaotic theory and chaotic techniques 10

2.2.1 Introduction 10

2.2.2 Standard chaotic techniques 14

2.2.3 Inverse approach 18

2.2.4 Approximation techniques 20

2.2.5 Phase space reconstruction 21

2.2.6 Summary 23

2.3 Support vector machine (SVM) 24

2.3.1 Introduction 24

2.3.2 Architecture of SVM for regression 26

2.3.3 Superiority of SVM over MLP and RBF Neural Networks 30

2.3.4 Issues related to model parameters 31

2.3.5 SVM for dynamics reconstruction of chaotic system 32

2.3.6 Summary 33

2.4 Conclusions 34

CHAPTER 3 SVM FOR PHASE SPACE RECONSTRUCTION 37

3.1 Introduction 37

3.2 Proposed SVM for dynamics reconstruction 38

3.2.1 Dynamics reconstruction with SVM 38

3.2.2 Calibration of SVM parameters 39

3.3 Proposed SVM for phase space and dynamics reconstructions 41

3.3.1 Motivations 41

3.3.2 Proposed method 42

3.4 Handling of large data record with SVM 43

3.4.1 Decomposition method 45

3.4.2 Linear ridge regression in approximated feature space 51

3.5 Summary and conclusion 59

CHAPTER 4 PARAMETER CALIBRATION WITH EVOLUTIONARY ALGORITHM 71

4.1 Introduction 71

4.2 Evolutionary algorithms for optimization 72

4.2.1 Introduction 72

4.2.2 Shuffled Complex Evolution 74

4.3 EC-SVM I: SVM with decomposition algorithm 79

4.3.1 Introduction 80

4.3.2 Calibration parameters 82

4.3.3 Parameter range 82

4.3.4 Implementation 85

4.4 EC-SVM II: SVM with linear ridge regression 87

4.4.1 Calibration parameters 87

4.4.2 Implementation 90

4.5 Summary 93

CHAPTER 5 APPLICATIONS OF EC-SVM APPROACHES 108

5.1 Introduction 108

5.2 Daily runoff time series 108

5.2.1 Tryggevælde catchment runoff 108

5.2.2 Mississippi river flow 109

5.3 Applications of EC-SVM I on daily runoff time series 111

5.3.1 EC-SVM I on Tryggevælde catchment runoff 111

5.3.2 EC-SVM I on Mississippi river flow 114

5.3.3 Summary 115

5.4 Applications of EC-SVM II on daily runoff time series 116

5.4.1 EC-SVM II on Tryggevælde catchment runoff 117

5.4.2 EC-SVM II on Mississippi river flow 118

5.5 Comparison between EC-SVM I and EC-SVM II 119

5.5.1 Accuracy 119

5.5.2 Computational time 119

5.5.3 Overall performances 120

5.6 Summary 121

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS 145

6.1 Conclusions 145

6.1.1 SVM applied in phase space reconstruction 146

6.1.2 Handling large data sets effectively 146

6.1.3 Evolutionary algorithm for parameters optimization 147

6.1.4 High computational performances 148

6.2 Recommendations for future study 148

REFERENCES 151

LIST OF PUBLICATIONS 162

SUMMARY

This research attempts to demonstrate the promising applications of a relatively new machine learning tool, support vector machine, on chaotic hydrological time series forecasting. The ability to achieve high prediction accuracy of any model is one of the central problems in water resources management. In this study, the high effectiveness and efficiency of the model is achieved based on the following three major contributions.

1. **Forecasting with Support Vector Machine applied to data in reconstructed phase space**. K nearest neighbours (KNN) is the most basic lazy instance–based learning algorithm and has been the most widely used approach in chaotic techniques due to its simplicity (local search). Analysis of chaotic time series, however, requires handling of large data sets which in many instances poses problems to most learning algorithms. Other machine learning techniques such as artificial neural network (ANN) and radial basis function (RBF) network, which are competitive to lazy instance-based learning, have been rarely applied to chaotic problems. In this study, a novel approach is proposed. The proposed approach implements Support Vector Machine (SVM) for the learning task in the reconstructed phase space and for finding the optimal embedding structure parameters based on the minimum prediction error. SVM is based on statistical learning theory. It has shown good performances on unseen data. SVM achieves a unique optimal solution by solving a quadratic problem and, moreover, SVM has the capability to filter out noise resulting from an ε-insensitive loss function. These special features lead SVM to be a better learning method than KNN

algorithm. SVM is able to capture the underlying relationship between forecasting and lag vectors more effectively.

2. **Handling large chaotic data sets effectively.** In the learning process, the forecasting task is a function of lag vectors. For cases with numerous training samples, such as in chaotic time series, the commonly used optimization technique in SVM for quadratic programming becomes intractable both in memory and in time requirement. To overcome the considerable computing requirements in large chaotic hydrological data sets effectively, two algorithms are employed: (1) Decomposition method of quadratic programming; and (2) Linear ridge regression applied directly in approximated feature space. Both schemes successfully deal with large training data sets efficiently. The memory requirement is only about 2% of that of the presently common techniques.
3. **Automatic parameter optimization with evolutionary algorithm.** SVM performs at its best when model parameters are well calibrated. The embedding structure and SVM parameters are simultaneously calibrated automatically with an evolutionary algorithm, Shuffled Complex Evolution (SCE).

In this study a proposed scheme, EC-SVM, is developed. EC-SVM is a forecasting **SVM** tool operating in the **C**haos inspired phase space; the scheme incorporates an **E**volutionary algorithm to optimally determine various SVM and embedding structure parameters. The performance of EC-SVM is tested on daily runoff data of Tryggevælde catchment and daily flow of Mississippi river. Significantly higher prediction accuracies with EC-SVM are achieved than other existing techniques. In addition, the training speed is very much faster as well.

NOMENCLATURE

τ	time delay
d	embedding dimension
k	number of nearest neighbours
$\mathbf{X}$	state vector in chaotic dynamical system
$\mathbf{y}$	lag vector in reconstructed phase space
$\mathbf{F}(\mathbf{X}_n)$	the evolution from $\mathbf{X}_n$ to $\mathbf{X}_{n+1}$
d_2	correlation dimension
$U(\cdot)$	unit step function
y	observation time series
$\mathbf{y}$	lag vector for reconstructed phase space
$I(\tau)$	average mutual information function
l	lead time for prediction
$\mathbf{x}$	input vector
y	target variable
y_o	observation value
N	training data size
n	dimension of input $\mathbf{x}$
$f(\mathbf{x})$	estimation function
$\varphi(\mathbf{x})$	feature vector corresponds to input $\mathbf{x}$
$\mathbf{w}$	weight vector for SVM
$E_{guarant}(\mathbf{w})$	guaranteed risk
CI	the confidence interval
h	VC dimension

L_ε	ε-insensitive loss function
ε	ε-insensitive parameter
$\xi^{(')}$	slack variables
J( )	Lagrangian function
$\alpha^{(')}$	Lagrange multiplier
K($\mathbf{x}$, $\mathbf{x}_i$)	inner-product kernel
$\mathbf{K}$	Kernel matrix
σ	width of Gaussian kernel function
C	trade off between empirical error and complexity of model
$\mathbf{y}_t$	input vector
y_{t+l}	l lead prediction
β	variable in standard quadratic programming of dual problem
β_S	working set
β_F	fixed variables
λ	Lagrange multiplier of standard quadratic programming
ϕ_j	eigenfunction of the integral equation
λ_j	eigenvalue of the integral equation
q	number of sub-samples
C′	ridge regression parameter
p($\mathbf{x}$)	probability density function in input space $\mathbf{x}$
$\mathbf{K}^{(q)}$	kernel matrix of q sample
$\mathbf{U}^i$	eigenvector matrix $\mathbf{K}^{(q)}$.
$\lambda_i^{(q)}$	eigenvalue of matrix $\mathbf{K}^{(q)}$
H_R	quadratic Renyi entropy
P	number of complexes

m	number of points in a complex
q	number of points in a sub-complex
p_{min}	minimum number of complexes required in population
α	number of consecutive offspring generated by a sub-complex
β	number of evolution steps taken by a complex
B	range of output data
$Q(t)$	runoff time series
$P(t)$	rainfall time series

LIST OF FIGURES

Figure 2.1 Illustration of data conversion from reconstructed phase space to feature space 35

Figure 2.2 Illustration of structural risk minimization 35

Figure 2.3 ε-insensitive loss function 36

Figure 2.4 Architecture of Support Vector Machine (SVM) 36

Figure 3.1 Reconstructed phase space data set with (τ =1, d=2, *l*=1) 61

Figure 3.2 Architecture of local model for dynamics reconstruction 61

Figure 3.3 Architecture of SVM for dynamics reconstruction 62

Figure 3.4 Diagram of dynamics reconstruction of chaotic time series 62

Figure 3.5 Schematic diagram of proposed SVM parameter set selection 63

Figure 3.6 Average mutual information (AMI) and time lag selection 64

Figure 3.7 Parameters determination and task performances with differences techniques: Standard, Inverse, and SVM approaches 64

Figure 3.8 Schematic diagram of SVM for phase space and dynamics reconstruction 65

Figure 3.9 Illustration of memory requirement for quadratic programming before and after decomposition scheme 66

Figure 3.10 SVM decomposition optimization problem with working set of 2 variables 66

Figure 3.11 Illustration of decomposition method in SVM quadratic programming 67

Figure 3.12 Illustration of shrinking process (reducing number of variables) in decomposition algorithm 68

Figure 3.13 Illustration of quadratic Renyi entropy function and scatter 69

Figure 3.14 Schematic diagram of ridge regression in feature space 70

Figure 4.1 Schematic diagram of Evolutionary Algorithms (EAs) 94

Figure 4.2 Search algorithm of Shuffled Complex Evolutions (SCE) 95

Figure 4.3 Basic processes in Competitive Complex Evolution (CCE): reflection and contraction 96

Figure 4.4 Proposed algorithm of EC-SVM I 96

Figure 4.5 Effect of varying C value on training time and test error: EC-SVM I 97

Figure 4.6 Effect of varying C value close to the output variable range B on training time and test error: EC-SVM I 98

Figure 4.7 Sensitivity of varying Kernel widths σ 99

Figure 4.8 Operational diagram of EC-SVM I 100

Figure 4.9 Distinction between unbiased distribution with large variance estimation (w) and biased distribution with small variance estimation (w^b) 101

Figure 4.10 Effect of varying C′ value on training time and test error: EC-SVM II 102

Figure 4.11 Effect of varying number of dimensions (q) of approximated features on training time and test and training errors: EC-SVM II 103

Figure 4.12 Effect of number of dimensions (q) on training time and test error: EC-SVM II 104

Figure 4.13 Operational diagram of EC-SVM II 105

Figure 4.14 Flow chart of the sub-modules in EC-SVM II 106

Figure 5.1 Location of Tryggevælde catchment, Denmark 122

Figure 5.2 Daily runoff time series of Tryggevælde catchment plotted in different time scales 123

Figure 5.3 Fourier transform and correlation dimension of daily Tryggevælde catchment runoff time series 124

Figure 5.4 Determination of time lag and embedding dimension: Tryggevælde catchment runoff time series 125

Figure 5.5 Location of Mississippi river, U.S.A. and runoff gauging station 126

Figure 5.6 Daily time series of Mississippi river flow plotted in different time scales 126

Figure 5.7 Fourier transform and correlation dimension of daily Mississippi river flow time series 128

Figure 5.8 Determination of time lag and embedding dimension: Mississippi river time series 129

Figure 5.9 Effect of C-range on number of iterations and training time: Tryggevælde catchment runoff time series 130

Figure 5.10 Computational convergence of EC-SVM I: Tryggevælde catchment runoff 130

Figure 5.11 Comparison between observed and predicted hydrographs using dQ time series in training: validation set of Tryggevælde catchment runoff 131

Figure 5.12 Effect of C range on number of iterations and training time of EC-SVM I: Mississippi rive flow 131

Figure 5.13 Computational convergence of EC-SVM I: Mississippi river flow 132

Figure 5.14 Comparison between observed and predicted hydrographs using dQ time series in training: validation set of Mississippi river flow 132

Figure 5.15 Scatter plot of EC-SVM II prediction accuracy using dQ time series: Tryggevælde catchment runoff 133

Figure 5.16 Scatter plot of EC-SVM II prediction accuracy using dQ time series: Mississippi river flow 133

Figure 5.17 Comparison between prediction accuracies resulting from EC-SVM I and EC-SVM II 134

Figure 5.19 Prediction accuracy and training time with dQ time series used in training: Tryggevælde catchment runoff 136

Figure 5.20 Prediction accuracy and training time with dQ time series used in training: Mississippi river flow 137

LIST OF TABLES

Table 4.1 Recommended SCE control parameters 107

Table 5.1 Range of parameters: EC-SVM I 138

Table 5.2 Training time and test error of EC-SVM I: Tryggevælde catchment runoff 138

Table 5.3 Optimal parameter set of EC-SVM I: Tryggevælde catchment runoff 138

Table 5.4 Prediction accuracy resulting from various techniques: Tryggevælde catchment runoff 139

Table 5.5 Training time and test error of EC-SVM I: Mississippi river flow 139

Table 5.6 Optimal parameter set of EC-SVM I: Mississippi river flow 140

Table 5.7 Prediction accuracy resulting from various techniques: Mississippi river flow 140

Table 5.8 Range of the parameters: EC-SVM II 141

Table 5.9 Training time and test error of EC-SVM II: Tryggevælde catchment runoff 141

Table 5.10 Optimal parameter set of EC-SVM II: Tryggevælde catchment runoff 141

Table 5.11 Prediction accuracy resulting from various techniques: Tryggevælde catchment runoff 142

Table 5.12 Training time and test error of EC-SVM II: Mississippi river flow 142

Table 5.13 Optimal parameter set of EC-SVM II: Mississippi river flow 143

Table 5.14 Prediction accuracy resulting from various techniques: Mississippi river flow 143

Table 5.15 Prediction accuracy of EC-SVM I and EC-SVM II 144

Table 5.16 Computation time of EC-SVM I and EC-SVM II 144

CHAPTER 1
INTRODUCTION

1.1 Background

Nature has been in observation for a very long time. From observations, we hope to better understand its system and the governing laws. Since physicists started research into the laws of nature, disorder, turbulent fluctuations, oscillation and ‘irregularity’ in nature have attracted the attention of many scientists. These ‘irregularity’ phenomena have simply been characterised as ‘noise’. The recent discovery of chaos theory changes our understanding and sheds new light on this type of nature study.

The first true experimenter in chaos was Lorenz, a meteorologist at MIT. In 1961 Lorenz derived the three ordinary differential equations describing thermal convection in a low atmosphere. He discovered that ever so tiny changes in climate could bring about enormous and volatile changes in weather. Calling it the Butterfly Effect, Lorenz pointed out that if a butterfly flapped its wings in Brazil, it could well produce a tornado in Texas (Hilborn, 1994).

Study on chaos has rapidly spread to various disciplines. It ranges from a flag snapping back and forth in the wind, the shape of the cloud and of a path of lighting, stock price rise and fall, microscopic blood vessel intertwining, to turbulence in the sea. Studies of chaotic applications on hydraulics and hydrology, however, started about 15 years or so ago and have shown promising findings.

Chaotic systems are deterministic in principle, e.g. a set of differential equations could describe the system under consideration. The system may display irregular time series. This irregularity of the system may, however, be mainly due to outside turbulence and yet, at the same time, the system is intrinsically dynamic. The system is very sensitive to the initial conditions, known as the butterfly effect. Initial conditions with any subtle difference will evolve into a totally different status as time progresses; therefore, a satisfactory prediction for a long lead-time is practically impossible for any such system. However, a good short-term prediction for the system is feasible.

Chaotic techniques analyse these irregular and sensitive systems. The embedding theory provides a means to transform the irregular time series into a regular system. The transformation is achieved when the original system is presented in the reconstructed phase space. The reconstructed phase space has a one-to-one relationship with the original system. A famous theorem is the Taken's theorem, which provides the lag vector approach to analyse the nonlinear dynamic system.

In the approach, two parameters (the time lag τ and the embedding dimension d) are to be determined. Various studies have been conducted in this domain. The commonly used techniques are the average mutual information (AMI), the false nearest neighbours (FNN), and the local model. The time lag τ can be determined by the AMI technique. The embedding dimension d is then determined after eliminating the false nearest neighbours using FNN technique.

The local model is commonly used for prediction. The local model typically adopts k nearest neighbours in the reconstructed phase space for interpolation to yield its prediction. Although it may be linear locally, globally it may be nonlinear.

For real time series, the embedding parameters obtained by these commonly used embedding techniques (AMI, FNN) may, as a matter of fact, not provide good prediction accuracy. This has triggered a series of studies (Casdagli, 1989; Casdagli et al., 1991; Gibson et al., 1992; Babovic et al., 2000a; Phoon et al., 2002; Liong et al., 2002) in the search for a more optimal set of τ and d. The studies showed that a search process through a set of combinations of τ and d provides better results than the standard chaotic technique.

In practice, prediction accuracy is often the most important objective. Using the prediction accuracy as a yardstick, Phoon et al. (2002) introduced an Inverse Approach whereby the optimal (d, τ, k) is first determined from forecasting and only then checked via the existence of the chaotic behaviour of the obtained embedding structure parameters, the (d, τ) set. The inverse approach was shown to yield higher prediction accuracy than the traditional approach. Most recently, Liong et al. (2002) replaced the brute force search engine in Phoon et al. (2002) with an evolutionary search engine, genetic algorithm (GA). Liong et al. (2002) showed that GA search engines not only allow a much more refined search in the given search space but also requires much less computational effort to yield the optimal (d, τ, k).

It should be noted that chaotic techniques are limited to the k nearest neighbour (KNN) learning algorithm to approximate the relationship between the lag vectors and the forecast variables. The restriction imposed to a limited k number of neighbours is to allow KNN be implemented in a large data record of chaotic time series. KNN algorithm is one of the oldest machine learning algorithms (Cover and Hart, 1967; Duda and Hart, 1973). A few new learning algorithms have been developed since then. These algorithms are very competitive and more powerful than KNN machine learning. The exploration of newly developed machine learning algorithms is still not widely implemented partly due to their difficulties in efficiently handling large data records.

1.2 Need for the present study

Other machine learning techniques such as artificial neural network (ANN) and radial basis function (RBF) network are competitors to the lazy instance-based learning KNN technique. However, they have been rarely explored and the exploration is limited to the dynamics reconstruction only. The phase space reconstruction techniques are still limited to the AMI and FNN traditional technique or KNN technique.

1.2.1 Support vector machine for phase space reconstruction

Support Vector Machine (SVM) is a relatively new machine learning tool (Vapnik, 1992). It is based on statistical learning and it is an approximate implementation of structural risk minimization which tolerates generalization on data not encountered during learning. It was first developed for classification problem and recently it has been successfully implemented in the regression problem (Vapnik et al., 1997).

SVM has several fundamental superiorities over ANN and RBF. First of all, one serious shortcomings of ANN is that the architecture of ANN has to be determined a priori or modified by some heuristic ways. The resulting structures of ANN are hence not optimal. The architecture of SVM, in contrast, does not need to be pre-specified before the training. Secondly, ANNs suffer the over-fitting problems. The way to overcome the over-fitting problem is rather limited. SVM is based on the structural risk minimization principle and the derivation is more profound. It considers both training error and confidence interval (capacity of the system). As a result, SVM has a good generalization capability (better performance on unseen data). Thirdly, ANNs can not avoid the risk of getting trapped in local minima while training due to its inherent formulation. SVM, on the other hand, solves a quadratic programming which has a unique optimal solution. Due to these attractive properties, SVM is regarded as one of the most well developed machine learning algorithms. Its applications are exceedingly encouraging in various areas.

So far, there has been no investigation on SVM applied to data in phase space reconstruction. Applying SVM on data mapped to the reconstructed phase space, where transformed data show clearer pattern, allows a technique such as SVM to perform a better forecasting task.

1.2.2 Handling large chaotic data sets efficiently

Chaotic time series analysis requires the efficient handling of a large data set. For most learning machine algorithms large data records require long computational times. KNN used as local model is dominant in chaotic techniques due to its simplicity. However, improvement in its prediction accuracy is desirable. Developing a SVM approach equipped with effective and efficient scheme to deal with large scale data sets is definitely much desirable for phase space reconstruction and forecasting.

The learning task approximates the forecast variables which is a function of lag vectors. When the number of training examples is large, say 7000, the currently used optimization technique for quadratic programming in SVM will become intractable both in memory and computational time requirement.

SVM's primal problem formulation is transformed into its dual problem in which Lagrange multipliers are the variables to be optimized. SVM solves the quadratic programming of 2N variables, where N is the size of training data set. The common technique of solving quadratic programming requires Hessian matrix, $O(N^2)$, to be stored in the memory. Chaotic time series analysis commonly requires large training data size N. The memory requirement is tremendously large and common PCs cannot afford such requirement. Moreover, the computational time is extremely expensive.

Existing publications on SVM applications for hydrological time series (Babovic et al., 2000b; Dibike et al., 2001; Liong and Sivapragasam, 2002) and dynamics reconstruction of chaotic time series analysis (Muller et al., 1997; Matterra and Haykin, 1999) revolve around those common techniques, e.g. Newton method, to solve the quadratic optimization problem. Small training set of about thousand records was used due to computational difficulty with Newton methods, e.g. 500 records in the work of Babovic et al. (2000b), 5 years daily data in Dibike et al. (2001), 3 years daily data in Liong and Sivapragasam (2002), 2,000 records in Muller et al. (1997). Only Matterra and Haykin (1999) investigated the impacts of different training sizes, up to 20,000 records, with supercomputers on prediction accuracy. Many hydrological daily time series come with 20-30 years or even longer records. The constraints posed thus far are the techniques used are not able to deal with large records efficiently. Thus, SVM equipped with the special algorithm which could effectively and efficiently deal with large scale data sets is highly desirable for phase space reconstruction and forecasting. Only such SVM can possibly provide high prediction accuracy in short computational time as well.

Recently there are some development of the special SVM scheme to deal large data size. The advanced SVM has not been noticed in areas of chaotic time series analysis and hydrological time series analysis. The exploration of the special SVM in chaotic hydrological time series analysis is extremely desirable.

1.2.3 Automatic parameter calibration

There are several parameters (C, ε, σ) in SVM which requires a thorough calibration. Parameter C controls the trade-off between the training error and the model complexity. Parameter ε is a parameter in the ε-insensitive loss function for empirical error estimation. The other parameter σ is a measure of the spread of the Gaussian kernel which influences the complexity of the model. Gaussian Kernel is a commonly employed Kernel in SVM and has been reported (Muller et al., 1997; Dibike et al., 2001; Liong and Sivapragasam, 2002) to generally provide good performances.

Currently there is no analytical way to determine the optimal values of these parameters. Only some rough guides are available in the literatures. The users are required to adjust the suggested parameter values. Adjustment task can be very time consuming. Thus, an automatic parameter calibration scheme is very much desirable.

1.3 Objectives of the present study

SVM is based on statistical learning theory and good performances on unseen data have been widely demonstrated. SVM achieves the unique optimal solution by solving a quadratic problem and, moreover, SVM has the capability to filter out noise resulting from ε-insensitive loss function. These special features of SVM lead to better learning than that of KNN algorithm. SVM is able to capture the underlying relationship between the forecast variables and the lag vectors more effectively.

This study focuses on establishing a novel framework with a relatively new powerful machine learning technique (SVM) to do forecasting on chaotic time series. This study first takes a close look at the possible applicability of SVM for chaotic data analysis. Combining its strength with the special feature of reconstructed phase space (mapping seemingly disorderly data into an orderly pattern) should be a more robust and yield higher prediction accuracy than traditional chaotic techniques.

Since there is a series of parameters (partially originating from SVM while others describing the system characteristics) required to be determined, a robust and efficient optimisation scheme such as Evolutionary Algorithms (EA) is considered to further enhance the proposed chaos based SVM scheme.

The objectives of this study can be specifically stated as follows:

1. To assess the performance and superiority of SVM over other traditional techniques in the analysis of chaotic time series;
2. To propose SVM regression model to the phase space reconstruction derived from the inverse approach;

3. To develop and implement advanced SVM equipped with effective and efficient scheme in handling large chaotic hydrological data sets;
4. To propose and implement an Evolutionary Algorithm to search for the optimal set for both the SVM and the embedding structure parameters;
5. To demonstrate the applications of the developed schemes on real hydrological time series and assess its performances. The performance of the proposed schemes will be compared with those of, for example, naïve forecasting, ARIMA, and other currently used chaotic techniques.

1.4 Thesis organization

Chapter 2 gives a brief overview of chaos theory, chaotic techniques and relevant optimisation schemes to derive the optimal embedding parameters. It also reviews Support Vector Machine and its applications in various disciplines.

Chapter 3 demonstrates how SVM in this study is applied to chaotic time series. It elaborates the proposed SVM approach applied in dynamics reconstruction and in phase space reconstruction. It also illustrates special schemes of SVM, introduced in this study, in handling large scale data sets. The proposed schemes require much less computational time and memory requirement.

Chapter 4 discusses the evolutionary algorithm (EA) used for parameters tuning. The basic idea of EA is described and the proposed schemes, EC-SVM I and EC-SVM II, are then demonstrated. Detailed implementations of EC-SVM I and EC-SVM II are presented.

Chapter 5 shows the applications of the proposed EC-SVM on daily Tryggevæld catchment runoff time series and Mississippi river flow time series. The prediction accuracy from the proposed EC-SVM I and EC-SVM II are compared with naive forecasting, ARIMA, and other currently used chaotic techniques.

Chapter 6 draws conclusions resulting from the current study and gives a number of recommendations for further research.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

Chaotic systems are not a rare phenomenon. Studies have shown that they exist widely in science, engineering and finance. In hydraulics, a good example of chaos is turbulence. Turbulent flow is irregular; however, for each flow particle we can write its governing equations, namely the Navier-Stokes equations and the mass conservation equation. Other examples of chaotic fluid motion are the weakly turbulent Couette-Taylor flow, Rayleigh-Benard convection. Similarly, chaotic phenomena have been observed in various hydrologic time series.

This chapter first reviews the basic ideas of chaos and chaotic techniques. In addition, more recent approaches in forecasting chaotic time series are reviewed. Review of Support Vector Machine (SVM), a relatively new machine learning tool (Vapnik, 1992; Vapnik et al., 1997), and its applications will follow.

2.2 Chaotic theory and chaotic techniques

2.2.1 Introduction

The precise definition of a chaotic system is shown in this subsection while the common identification of correlation dimension and embedding theorem are described in the following subsections.

(1) Definition of Chaos

Chaos refers to the irregular, unpredictable behaviour observed in a dynamic system that is extremely sensitive to small variations in initial conditions, known as the butterfly effect (Lorenz, 1963). It is a deterministic system but with complex behaviour.

A dynamic system is a system which continuously evolves with time and can be determined by knowledge of its past history. Mathematically, the time evolution of state variables is expressed as:

$$\mathbf{X}_{n+1} = \mathbf{F}(\mathbf{X}_n) \tag{2.1}$$

There are three major issues in the description of a dynamical system: (1) the phase space; (2) the dynamical rule; and (3) the initial value. The phase space or state space, with its coordinates, describes the dynamical state. An orbit (or trajectory) is the path of a solution in the space. A dynamical rule specifies the immediate future trend of all state variables, e.g. Eq. (2.1) describes the evolution from $\mathbf{X}_n$ to $\mathbf{X}_{n+1}$. For a given initial condition the solution of a chaotic system is unique. This is in contrast to the 'stochastic' or 'random' system where more than one consequence is possible.

The sensitivity of chaotic system to its initial condition can be expressed in the following way:

For any $\varepsilon > 0$, and for some $r > 0$, for each $\mathbf{X}_0$ in the set S, there is a $\mathbf{X}'_0$ such that $|\mathbf{X}_0 - \mathbf{X}'_0| < \varepsilon$ provides $|\mathbf{X}_n - \mathbf{X}'_n| > r$ after some n steps evolution.

For a fixed distance r, no matter how precise one specifies an initial condition, there are points nearby this initial state that will be separated by a distance away after n steps. This means that, as time goes on, any tiny difference will grow rapidly and become significant.

Another characteristic of chaotic systems is its irregularity and unpredictability. The irregularity is the intrinsic property of a dynamic system and it is not originated from outside influences. As a consequence of the long-term unpredictability, time series generated from chaotic systems may appear to be irregular and disordered. However, chaos is not completely disordered and is feasible for short-term prediction.

Chaotic time series typically provide a low value dimension even though they appear quite irregular and have a broad band power spectrum. Usually, the chaos attractor is fractal. The fractal dimensions characterise the geometric figure of the attractor. Fractal has come to mean any system that displays the attribute of self-similarity. No matter how closely you look at a fractal, there is, so to say, no straight line in it.

The dimension of the attractor is one of the measures to distinguish the chaotic time series from the stochastic time series. Box counting dimension is one of the ways for computing the fractal dimensions. If the phase space is covered with small *k*-dimension cubes with edge ε, the orbit is visiting each of these cubes in turn. The fractal dimension can be defined as:

$$D = \lim_{\varepsilon \to 0} \frac{\ln M(\varepsilon)}{\ln(1/\varepsilon)} \tag{2.2}$$

where M(ε) : minimum number of such cubes needed to cover the set.

(2) Identifications

There are three major characterisations of chaotic system: (1) Lyapunov exponents characterise the stretching properties of the trajectory under the process of evolution (e.g., Wolf et al., 1985; Eckmann et al., 1986); (2) the fractal dimensions characterise the geometric figure of the attractor (e.g., Grassberger and Procaccia, 1983a, b; Termonia and Alexandrowicz, 1983; Theiler, 1987); (3) the Kolmogorov entropy characterises the complexity of the trajectory structure (e.g., Grassberger and Procaccia, 1983c).

Chaotic systems do not necessarily require the existence of a positive Lyapunov exponent. A positive Lyapunov exponent is observed for random processes (Rodriguez-Iturbe et al., 1989; Jayawardena and Lai, 1994).

The correlation dimension (D_2) can be easily determined from the experimental data and is commonly used for identification of the chaotic system. The basic idea was suggested by Grassberger and Procaccia (1983a, b). For a given data set on the attractor: $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$,

$$\hat{C}(\varepsilon) = \lim_{n\to\infty}\left[\frac{1}{n^2}\sum_{j=1}^{n}\sum_{i=1}^{n} U(\varepsilon - \left\|\mathbf{z}_i - \mathbf{z}_j\right\|)\right] \tag{2.3a}$$

where U(·) is unit step function, i.e. U(x) =1, x>0; and U(x)=0, x≤0. Correlation dimension D_2 is then can be calculated as:

$$D_2 = \lim_{\varepsilon\to 0}\left[\frac{\ln \hat{C}(\varepsilon)}{\ln \varepsilon}\right] \tag{2.3b}$$

(3) Embedding theory

Embedding theory (Takens, 1981; Sauer et al., 1991) provides a theoretic foundation to chaotic analysis from experimental data. With observation data, it is possible to detect the evolution of the system and to reconstruct the chaotic attractor on the basis of the embedding technique.

Theorem 1 (Whitney Embedding Existence Theorem) Let A be a compact smooth manifold of dimension d in R^k. Almost every smooth map $R^k \rightarrow R^{2d+1}$ is an embedding of A. m > 2d can be regarded as the necessary condition for F(A) not to intersect with itself.

Theorem 2 (Fractal Whitney Embedding Prevalence Theorem): Let A be a compact subset of R^k of box counting dimension D_0, and n an integer such that $n>2D_0$. For almost every smooth map F: $R^k \rightarrow R^n$,

1. F is one-to-one on A
2. F is an immersion on each compact subset C of a smooth manifold contained in A.

The famous Taken's time delay-embedding theorem is as follows:

Given a delay time τ, a time lag vector **y** of d dimensions can be defined as:

$$\mathbf{y}_t = \left(y_t, y_{t-\tau}, \ldots, y_{t-(d-1)\tau}\right) \tag{2.4}$$

If d is large enough, then the mapping between lag vector (**y**) and state variable (**X**) is smooth and invertible. The study of observation **y** is also the study of the solutions **X** of the underlying dynamic system.

2.2.2 Standard chaotic techniques

A time series is often characterised as chaotic time series, typically with low value correlation dimension and a broad band spectrum from Fourier transform. Two major reconstructions are involved, i.e. phase space reconstruction in normal Euclidian space and dynamics reconstruction. The phase space reconstruction determines the appropriate time delay and embedding dimension. Several standard chaotic techniques can be used to select time lag and embedding dimension. The forecasting can be subsequently carried out by fitting a function relating the lag vectors and the predicted variables.

(1) Time lag selection

Mees et al. (1987) suggested a time lag at which the autocorrelation function first crosses zero. Other approaches consider a delay time at which the autocorrelation function attains a certain value; say 0.1 (Tsonis and Elsner, 1988), or 0.5 (Schuster, 1988). Fraser and Swinney (1986) suggested using average mutual information (AMI) as a nonlinear correlation function to determine the required time lag. For a set of measurements, y(n), the mutual information between y(n) and y(n+τ) is defined by:

$$I(\tau)=\sum_{y(n);y(n+\tau)} P\big(y(n),y(n+\tau)\big)\log_2\left[\frac{P(y(n),y(n+\tau))}{P(y(n))P(y(n+\tau))}\right] \qquad (2.5)$$

P(y(n)) is an individual probability and P((y(n), y(n+τ)) is a joint probability. It can be seen that I(τ) is greater than zero. As τ gets significantly large, the chaotic signals y(n) and y(n+τ) become independent from each other. The joint probability becomes the product of the individual probabilities as shown in Eq. (2.6a):

$$P\big(y(n),y(n+\tau)\big)=P\big(y(n)\big)P\big(y(n+\tau)\big); \qquad (2.6a)$$

$$\log_2 1 = 0 \qquad (2.6b)$$

Thus, I(τ) tends to go to zero as τ gets large. The τ-value at the first minimum of I(τ) is commonly suggested to be chosen as the time lag. Abarbanel (1996) proposed a method to form histogram from the sample data to estimate I(τ).

(2) Embedding dimension selection

According to the embedding theorem of Takens (1981), to characterize a dynamic system with an attractor dimension d_2, a $d \geq 2d_2+1$ dimensional phase space is adequate to undo the overlaps. Abarbanel et al. (1990), however, suggested that an embedding dimension just greater than the attractor dimension is sufficient. Kennel et al. (1992) developed the False Nearest Neighbour (FNN) method to choose embedding dimension.

The basic idea is that if the embedding dimension is d, then the neighbour points in P^d are also the neighbour points in P^{d+1}. If this is not the case these points are then called false neighbour points. If the number of the false neighbour points is negligible then this d can be chosen as the embedding dimension.

A lag vector $\mathbf{y}_t$ in d dimensions has its nearest neighbour point $\mathbf{y}_t'$. The Euclidean distance $R_d(t)$ can be used as a measure of the distance between these two points:

$$R_d(t)^2 = \sum_{n=1}^{d}\left[y(t-(n-1)\tau) - y'(t-(n-1)\tau)\right]^2 \tag{2.7}$$

If the dimension increases by one, to d+1 dimension, the lag vector is:

$$y_t^{d+1} = [y(t), y(t-\tau), ..., y(t-(d-1)\tau), y(t-d\tau)] \tag{2.8}$$

The Euclidean distance $R_{d+1}(t)$ between the points $\mathbf{y}_t$ and $\mathbf{y}_t'$ is:

$$\begin{aligned} R_{d+1}(t)^2 &= \sum_{n=1}^{d+1}\left[y(t-(n-1)\tau) - y'(t-(n-1)\tau)\right]^2 \\ &= R_d(t)^2 + \left|y(t-d\times\tau) - y'(t-d\times\tau)\right|^2 \end{aligned} \tag{2.9}$$

Empirically, if the additional distance $\left|y(t-d\times\tau) - y'(t-d\times\tau)\right|$ relative to the Euclidean distance $R_d(t)$

$$\frac{\left|y(t-d\times\tau) - y'(t-d\times\tau)\right|}{R_d(t)} \tag{2.10}$$

is greater than a threshold value of approximately 15, these two points are false neighbours. This number of 15 is an experimental value. It may change due to the nature of the sample data set.

(3) Prediction

The popularly used delay coordinates reconstruction technique reproduces the set of dynamical states of a system, using the lag vector, from the measured time series. Prediction is one of the applications of dynamics reconstruction. The lag vector has a one-to-one mapping to the state variable of the dynamic system and the evolution of the lag vector follows that of the state variable (Farmer and Sidorowich, 1987). The evolution of **y** of can be written as:

$$\mathbf{y}(t+1) = \mathbf{F}(\mathbf{y}(t)) \tag{2.11}$$

The local model considers a local function $\boldsymbol{f_L}$ for each local region. Usually each region covers several nearest neighbour points in the data set. This set of $\boldsymbol{f_L}$ builds up the approximation of the **F** for the whole domain. The first component of the above equation is what we need for the prediction of $y(t+1)$:

$$y(t+1) = \mathrm{F}_1(\mathbf{y}(t)) \tag{2.12}$$

K number of nearest neighbours of $\mathbf{y}$(t) in the reconstruction space, i.e. points with the smallest Euclidean space in R^d, denoted as $\mathbf{y}_i'(t)$, i=1,2,…,k is required. This is followed by the construction of a local predictor f_{L1} in the region of these k nearest neighbours. A linear interpolation is carried out, which results in the following predictor:

$$y(t+1) = \alpha_0 + \sum_{n=1}^{d} \alpha_n y(t-(n-1)\tau) \tag{2.13}$$

For k = d+1, this is equivalent to a linear interpolation and sufficient to determine the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_d$. It is often suggested to use k > d+1 to ensure the stability. It has been shown that zero-th order and first order interpolation provide a reasonably good fitting. Higher order polynomials may not provide significantly better results than polynomial of first order (Farmer and Sidorowich, 1987; Zaldívar et al., 2000).

Many studies on chaos in meteorological and hydrological time series follow the above standard chaotic techniques. (e.g., Nicolis and Nicolis, 1984; Fraedrich, 1986, 1987; Grassberger, 1986; Essex et al., 1987; Hense, 1987; Tsonis and Elsner, 1988; Rodriguez-Iturbe et al., 1989; Sharifi et al., 1990; Islam et al., 1993; Jayawardena and Lai, 1994; Porporato and Ridolfi, 1996, 1997, Sivakumar et al., 1998; Zaldívar et al., 2000)

2.2.3 Inverse approach

Casdagli (1989) first proposed an inverse approach to construct a robust predictive model directly from time series data. The study showed the effect of embedding dimension using brute force search while the other two prediction parameters (time delay and the number of nearest neighbours) were selected following some recommendations. The author studied different theoretical time series from low to high dimensional chaos. Casdagli et al. (1991) conducted a detailed study on state space reconstruction in the presence of noise for predicting time series. Gibson et al. (1992) focused on the advantage of using prediction accuracy as a useful criterion for practical state space reconstruction.

Babovic et al. (2000a) implemented an inverse approach to produce prediction parameters from a wide range of values of the embedding dimension, the delay time and the number of nearest neighbours. A Genetic Algorithm (GA) was employed to search for the optimal values of the embedding parameters (d, τ, k). They divided the data into two sets, state space reconstruction set and the production set. The values of the parameter set (d, τ, k) are optimal when the prediction error is minimum. A local model is used in the study to do a l-lead day prediction. Thus, the set (d, τ, k) which yields the least l-lead day prediction error is the optimal set. They applied the proposed approach on water level prediction of Venice Lagoon, Italy. The study shows that the prediction accuracy, on the production set, is improved by 20% to 35% compared to that resulting from the standard approach.

Phoon et al. (2002) also searched for the optimal embedding parameters which yield the highest prediction accuracy. Phoon et al. (2002) dealt also with two other issues: (1) would the resulting optimal parameter set (d,τ, k) be dependent on the lengths of both state space reconstruction and calibration sets?; and (2) would the resulting optimal set (d, τ, k) demonstrate the chaotic behaviour? In their approach, the time series is divided into three subsets, i.e. state space reconstruction set, calibration set, and production set. The calibration set is used to check the performance of the embedding structure parameter set proposed from the state space reconstruction set. The resulting (d, τ) set is then checked whether the set demonstrates the chaotic behaviour. A brute force search engine is used in their study. With the range and incremental step of each of the parameters considered, a total number of 4104 evaluations are required. They applied the approach first on a noise-free Mackey-Glass time series and then on a daily runoff of Tryggevaelde catchment. Higher prediction accuracy was achieved by the inverse approach than the standard approach.

Liong et al. (2002) analyzed the same problem as that in Phoon et al. (2002) with, however, two main differences: (1) a genetic algorithm (GA) search engine is employed; and (2) a constant and smallest incremental step of 1 is adopted for each of the parameters considered. The study shows that GA search engine not only yields higher prediction accuracy but also with a much less number of evaluations. Their prediction accuracy is higher than that of Phoon et al. (2002).

2.2.4 Approximation techniques

The most conceptually easily accepted approximation algorithm is the polynomial predictor. It fits **F$_l$** using an m-th order polynomial in d dimensions. Thus, it deals with a polynomial with $\binom{m+d}{m}$≡(m+d)!/(m!d!) ≅ d^m parameters. As the range of m and d values increase, the number of free parameters gets larger as well. Also when the training size is large, it causes a storage problem. There is no solid guideline to select appropriate polynomial order. It is known that polynomials of high orders tend to yield undesirable oscillation.

K nearest neighbours (KNN) is the most basic instance-based learning method. It is widely used in chaotic techniques due to its simplicity for the learning algorithm on large data sets. The main requirement is that the data set must be very dense at every point and the number of neighbour points at least be d+1 so that the local coefficients can be estimated as given in Eq. (2.13). For real world data it may be too demanding. Moreover, a local model is discontinuous from neighbourhood to neighbourhood.

Artificial Neural Networks (ANNs) have shown powerful approximation abilities, in particular, after the discovery of the back propagation training algorithm in the mid-1980s. Casdagli (1989) proposed the Artificial Neural Network (ANN) and Radial basis functions (RBF) to approximate the chaotic system. RBF is another type of instance learning and global interpolation technique with good localization properties. The 'optimal' structure of ANN and RBF, i.e. number of the hidden layers, number of hidden neurons, and the centres of the RBFs, has to be determined by the user through a trial-and-error approach. It should be noted that the resulting 'optimal' set may not be the global optimum.

Support Vector Machine (SVM) is a relatively new learning algorithm (Vapnik, 1992; Vapnik, et al., 1997). Muller et al. (1997) employed SVM for chaotic time series forecasting. They proposed to use SVM on artificial noise mixed Makey-Glass and Santa Fe time series prediction. Since SVM obtains its optimal structure itself during training, it does not suffer from the 'optimal' structure selection. SVM improves the results, obtained from the neural network, by 29% with ε-insensitive loss function. A satisfactory performance was shown. Mattera and Haykin (1999) employed SVM on dynamics reconstruction of a chaotic system. They applied SVM on noise-free and noisy Lorenz time series reconstruction. The results showed the effectiveness of SVM in performing the nonlinear reconstruction. SVM is largely insensitive to measurement noise.

2.2.5 Phase space reconstruction

The concept of lag vector is not only used in chaotic time series. On the contrary, the popularly used ARMA models also use the lag vector; and most of the ANN applications also use time lag as input layer. Auto-Regressive and Moving Average (ARMA) is the most traditional technique for time series analysis. It describes the time series as a linear function of p previous data and q previous white noise process, i.e. ARMA (p, q):

$$x_{t+1} = a_0 x_t + a_1 x_{t-1} + \ldots + a_p x_{t-p} + a_{p+1} + b_0 \eta_t + b_1 \eta_{t-1} + \ldots + b_q \eta_{t-q} \tag{2.14}$$

ARMA expresses the future rainfall/runoff, for example, as a linear function of past data in hydrological time series analysis. The selections of the proper order of p, q are mainly based on empirical identification of the 'cut off' or 'dying down' pattern of the sample autocorrelation function, and sample partial autocorrelation function respectively. There are two major questions: (1) The future rainfall/runoff may be not a linear function of the past data; (2) The dependence on the previous data could be of other possibilities instead of time lag of 1 only (such as that shown in Eq. (2.14), i.e. each of the following time lags of 2, 3, 4, etc. could be a possibility.

Recently there have been several nonlinear regression models developed for time series analysis. Neural network is one of most popular techniques in dealing with the nonlinear relationship. For runoff forecasting, the input layer mainly contains previous data of rainfall, temperature, and runoff, for example, of a 'window size' d (Karunanithi et al., 1994; Zealand et al., 1999; Toth et al., 2000; Anctil et al., 2004). Recently Support Vector Machine (SVM) application for hydrological time series forecasting also follows the above approach (Babovic et al., 2000b; Dibike et al., 2001; Liong and Sivapragasam, 2002). Almost all ANN and SVM applications on rainfall or runoff forecasting fixed their selected time lag at 1 and did not investigate other time lags. Some studies also fixed the window size d.

In chaotic technique, the future rainfall/runoff is a function of the lag vectors. The proper lag vector is chosen among various different time lags and embedding dimensions. i.e.:

$$x_{t+1} = f(x_t, x_{t-\tau}, \ldots x_{t-(d-1)\tau}) \tag{2.15}$$

As it can be seen from Eq. (2.15), that the above description includes the ARMA and the existing ANN applications. In ARMA, the time lag is fixed at 1 and the embedding dimension is p; the resulting model is fitted by a linear function. In ANN applications, the time lag is fixed as 1 and the embedding dimension is the 'window size'.

Most of the chaotic applications show that the optimal time lag could be other values besides a time lag of 1. Optimal time lags for rainfall/runoff time series reported have been 1, 2, 3, 40 (Phoon et al., 2002); and 3, 6, 9 (Doan et al., 2003) for daily runoff time series.

The regression ANN model can be viewed as a multivariate embedding technique. Similarly, proper time lag and embedding dimension should be optimally determined.

2.2.6 Summary

The discovery of chaos theory and accurate short-term predictions in many seemingly irregular natural and physical processes has triggered a series of research works in the field of water resources, especially in hydrology.

The concept of phase space reconstruction is a very valuable contribution to the time series analysis. The information obtained would render better choice of input neurons in ANN, for example.

In the AMI method, choosing the time delay τ when $I(\tau)$ arrives at its first minimum is suggested. It should be noted that there is no strong theoretical support to this prescription. In addition, the proposed time delay gives no guarantee of good forecasting results. A Similar problem occurs in the false nearest neighbour approach in determining the embedding dimension d. A threshold value to determine whether the considered points are false nearest neighbours is empirically derived for some chaotic systems. It is thus not to be expected that all real time series will follow that empirically selected threshold value. A change in the threshold value will affect the embedding dimension, d.

Recently a series of attempts (Casdagli, 1989; Casdagli et al., 1991; Gibson et al.,1992; Babovic et al., 2000a; Phoon et al., 2002; Liong et al., 2002) using the inverse approach has been offered. There the objective is to find the optimal (d, τ, k) set which gives the minimum prediction error. Results showed that their prediction accuracy is, as expected, much higher than that resulting from the traditional chaotic techniques.

Local models are widely used in traditional chaotic techniques due to their simplicity in the learning algorithm for large data sets. This simplicity renders the local model superior over other existing learning techniques prior to the arrival of SVM. It is therefore of interest to thoroughly explore the performance of SVM in chaotic time series and compare its performance with its forerunners.

2.3 Support vector machine (SVM)

Support Vector Machine (SVM) is a relatively new machine learning tool. It is regarded as one of the most elegant and promising learning techniques developed thus far. It is an approximate implementation of structural risk minimization which tolerates generalization on data not encountered during learning. Recently, SVM has attracted the attention of many researchers. It has been successfully implemented in the regression problem and its performances are quite encouraging (e.g. Müller et al., 1997; Liong and Sivapragasam, 2002).

2.3.1 Introduction

The SVM algorithm has been developed over the last three decades. In its present form, however, SVM was only recently developed at AT&T Bell Laboratories by Vapnik and co-workers first for classification problems (1992) and later for regression problems (1997). It is grounded in the framework of statistical learning theory, or VC theory. SVM has become competitive with the best available learning machine algorithms shortly after it was developed.

Let's consider a training data set ($\mathbf{x}_i$, y_{oi}), i = 1, 2, …, N where $\mathbf{x}$ is the input vector with dimension n, and y_o is the corresponding output with dimension of 1. The regression model is to estimate $f(\mathbf{x})$:

$$y_o = f(\mathbf{x}) + v \tag{2.16}$$

$f(\mathbf{x})$ is the conditional expectation E[D|$\mathbf{x}$] with D as the random variable and a realization of y_o. v is the adaptive noise term. The estimation of y_o is denoted by y. In SVM, the input space $\mathbf{x}$ is transformed to a higher dimension space $\varphi(\mathbf{x})$, Fig. 2.1. $\{\varphi_j(\mathbf{x})\}$ refers to the hidden space or feature space. These nonlinear basis functions $\{\varphi_j(\mathbf{x})\}$ convert the original non-linear complex function $f(\mathbf{x})$ into a linear equation in the feature space so that y can be linearly described:

$$y = \textstyle\sum_{i=0}^{m} w_i \varphi_i(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) \tag{2.17}$$

A small training error does not, however, guarantee that a small error will result in unseen data. The performance on the unseen data is termed "generalization". Structural risk minimization principle (SRM) considers the fundamental issue of how to control the generalization ability mathematically, Fig. 2.2. According to SRM, the generalization error is lower than a guaranteed risk defined as:

$$E_{guarant}(\mathbf{w}) = E_{train}(\mathbf{w}) + CI(N,h) \tag{2.18}$$

CI is the confidence interval which is a function of training size and VC dimension h. VC dimension is a purely combinatorial concept that has no connection with the geometric dimension and is a measure of the capacity of the learning machine. For example,

$$y = \textstyle\sum_{i=1}^{m} w_i x_i + b \tag{2.19}$$

has a VC dimension of (m+1). The training error decreases as the capacity or h increases while the confidence interval increases. The method of the structure minimization is to find the best compromise between the training error and the confidence interval (i.e. complexity of the approximation function).

2.3.2 Architecture of SVM for regression

A quadratic loss function is popularly used in neural networks, i.e. in multilayer perceptrons and radial-basis function networks due to its computational convenience. However, it is quite sensitive to the presence of the outliers. It performs poorly when the underlying distribution of the additive noise has a long tail. To overcome this limitation, SVM adopts an ε-insensitive loss function. This would allow the model to become more robust, i.e. insensitive to small changes.

$$L_\varepsilon(y_o, y) = \begin{cases} |y_o - y| - \varepsilon, & \text{for } |y_o - y| \geq \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

Figure 2.3 illustrates the dependence of L_ε (y_o, y) on the error (y_o - y). If the deviation is less than ε, L_ε (y_o, y) is equal to zero. Only when the deviation is larger than ε, the error is considered. The model can filter noisy data with respect to an ε-deviation. The issue then revolves around the following optimal problem:

Minimizing the empirical risk: $R_{emp} = \frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(y_{oi}, y_i)$ (2.21a)

Subject to the inequality : $\|\mathbf{w}\|^2 \leq c_0$ (2.21b)

The constraint in Eq. (2.21b) reflects the complexity degree of the model. The higher $\|\mathbf{w}\|^2$ is, the more complex is the model. As always, a less complex model is preferred. Introducing the following slack varibles, ξ_i, ξ_i' (Fig. 2.3), defined as:

$$\text{if } y_o - y \geq \varepsilon, \qquad \xi = y_o - y - \varepsilon \geq 0, \xi' = 0 \tag{2.22a}$$

$$\text{if } y_o - y \leq -\varepsilon, \qquad \xi' = y - y_o - \varepsilon \geq 0, \xi = 0 \tag{2.22b}$$

the ε-insensitive loss function can then be reformulated as:

$$y_{oi} - \mathbf{w}^{\mathrm{T}}\varphi\,(\mathbf{x}_i) \leq \varepsilon + \xi_i \tag{2.23a}$$

$$\mathbf{w}^{\mathrm{T}}\varphi\,(\mathbf{x}_i) - y_{oi} \leq \varepsilon + \xi_i' \tag{2.23b}$$

$$\xi_i \geq 0\,, \xi_i' \geq 0\,, i = 1, 2, \ldots, N \tag{2.23c}$$

The constrained optimization problem in Eq. (2.21) can be viewed as a cost minimization function problem:

$$\Phi(\mathbf{w}, \xi, \xi') = C\left(\textstyle\sum_{i=1}^{N}(\xi_i + \xi_i')\right) + \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{2.24}$$

subject to the constrains in Eq. (2.23). The constant C is a user specified parameter. The Lagrangian function can now be defined as:

$$\begin{aligned} J(\mathbf{w}, \xi, \xi', \alpha, \alpha', \gamma, \gamma') = C\textstyle\sum_{i=1}^{N}(\xi_i + \xi_i') + \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\alpha_i[\,\mathbf{w}^{\mathrm{T}}\varphi\,(\mathbf{x}_i) - y_{oi} + \varepsilon + \xi_i\,] \\ - \textstyle\sum_{i=1}^{N}\alpha_i'[y_{oi} - \mathbf{w}^{\mathrm{T}}\varphi\,(\mathbf{x}_i) + \varepsilon + \xi_i'] - \sum_{i=1}^{N}(\gamma_i\xi_i + \gamma_i'\xi_i') \end{aligned} \tag{2.25}$$

The saddle points of Eq. (2.25) can be obtained by setting differential of J equal to zero:

$$\frac{\partial J}{\partial \mathbf{w}} = 0\,, \frac{\partial J}{\partial \xi_i} = 0\,,\ \frac{\partial J}{\partial \xi_i'} = 0 \tag{2.26}$$

which in turn will yield:

$$\mathbf{w} = \textstyle\sum_{i=1}^{N}(\alpha_i - \alpha_i')\,\varphi\,(\mathbf{x}_i),\ \gamma_i = C - \alpha_i\,,\ \gamma_i' = C - \alpha_i' \tag{2.27}$$

Introducing Eq. (2.27) into Eq. (2.25) poses a newly formulated optimization problem:

$$\begin{aligned} \text{Maximize:} \quad Q(\alpha, \alpha') = \textstyle\sum_{i=1}^{N} y_{oi}(\alpha_i - \alpha_i') - \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i') \\ - \frac{1}{2}\textstyle\sum_{i=1}^{N}\sum_{i=1}^{N}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{2.28a}$$

Subject to : $\sum_{i=1}^{N}(\alpha_i - \alpha_i') = 0$ (2.28b)

$0 \le \alpha_i \le C, 0 \le \alpha_i' \le C$, i =1, 2,…, N (2.28c)

where $K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$ is the inner-product kernel. The objective is to solve the dual problem by maximizing the $Q(\alpha,\alpha')$ as stated in Eq. (2.28). The dual problem is a quadratic programming which involves maximization of a quadratic function subject to a linear constraint. After finding the optimal Lagrangian multipliers, the weights are then determined through Eq. (2.27). Only those data points with $\alpha_i \neq \alpha_i'$ are the support vectors. The number of the support vectors (N′) is usually much smaller than the sample size N originally given. The regression function f is:

$$y = f(\mathbf{x}) = \mathbf{w}^T\varphi(\mathbf{x}) = \sum_{i=1}^{N}(\alpha_i - \alpha_i')K(\mathbf{x},\mathbf{x}_i) \quad (2.29)$$

It should be noted that an inner product kernel function chosen for SVM, K($\mathbf{x}$, $\mathbf{x}_i$), must satisfy Mercer’s theorem (Mercer, 1908; Courant and Hilbert, 1970).

Mercer’s Theorem: Let $K(\mathbf{x}, \mathbf{x}')$ be a continuous symmetric kernel that is defined in the closed interval $\mathbf{a} \le \mathbf{x} \le \mathbf{b}$ and $\mathbf{a} \le \mathbf{x}' \le \mathbf{b}$, i.e.

$$\lambda\phi(\mathbf{x}') = \int_c K(\mathbf{x},\mathbf{x}')\phi(\mathbf{x})d\mathbf{x}, \quad (2.30)$$

$K(\mathbf{x}, \mathbf{x}')$ can be expanded as a series:

$$K(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{\infty}\lambda_i\phi_i(\mathbf{x})\phi_i(\mathbf{x}') \quad (2.31)$$

with positive coefficients, $\lambda_i > 0$. For this expansion to be valid and to converge absolutely and uniformly, it is necessary and sufficient that $K(\mathbf{x}, \mathbf{x}')$ is positive definite, i.e.:

$$\int_b^a\int_b^a K(\mathbf{x},\mathbf{x}')f(\mathbf{x})f(\mathbf{x}')d\mathbf{x}d\mathbf{x}' \ge 0 \quad (2.32)$$

holds for all $f(\cdot)$ that $\int_a^b f^2(\mathbf{x})d\mathbf{x} < \infty$.

Mercer's theorem can only verify whether a proposed kernel is actually an inner-product kernel. Several kernel functions which satisfy the Mercer's theorem are available. They are, for example,

1. Dot product kernel function: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{T}\mathbf{x}'$

2. Polynomial kernel function: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^{T}\mathbf{x}'+1)^{p}$

3. Gaussian kernel function: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}'\|^2\right)$

4. Sigmoid kernel function: $\tanh(\beta_0\,\mathbf{x}^{T}\mathbf{x}'+\beta_1)$ only for some values of β_0 and β_1.

As indicated, sigmoid kernel function is somewhat restricted. In the case of dot product, the feature space is actually the original variable and it may not be sufficient for real time series application with nonlinear relationship. The power of the dot product kernel function is far too limited. Polynomial kernel is more powerful than dot product kernel. However, polynomials of high degree have undesirable oscillation. Many studies (Babovic et al., 2000b; Dibike et al., 2001; Liong and Sivapragasm, 2002) have shown that Gaussian kernel has demonstrated a good performance in hydrology. This is actually to be expected since the dimension of the feature space of Gaussian kernel is infinite and it has a powerful feature to approximate nonlinear relationships. The parameter σ controls the complexity of the model. The smaller the σ value is, the more powerful the Gaussian kernel is.

As shown in Fig. 2.4, SVM has a very similar structure to that of a radial basis neural network function. It can be viewed as a one-layer machine. For an input vector $\mathbf{x}$ with n dimensions, there are N′ support vectors for the machine, the structure of the machine can be represented as in Fig. 2.4.

2.3.3 Superiority of SVM over MLP and RBF Neural Networks

SVM is an elegant and highly principled learning method. Its derivation follows the principle of structural risk minimization which makes the derivation more profound. Multilayer Perceptron (MLP) and radial basis function (RBF) networks are probably the most popular nonlinear estimation techniques. Both MLP networks and RBF networks have several known major drawbacks such as:

1. The architecture of MLP, i.e. the number of hidden layers and hidden neurons, has to be determined a priori or modified while training by some heuristic ways. RBF networks have to choose the number of RBF functions and the centres of those RBFs. The resulting structures from these heuristic approaches on both MLP and RBF are not necessarily optimal. SVM can be viewed as one layer machine. The architecture of SVM does not need to be specified before training. During training, SVM determines the support vectors itself and those support vectors act as neurons. Therefore SVM, unlike MLP and RBF, does not suffer the architecture determination problem.
2. In order to fit the training data from a nonlinear system, the learning machine must be powerful enough to detect those nonlinear complex relationships. Therefore the over-fitting problem can not be avoided. ANNs suffer from the over- fitting problem and the way to overcome the over-fitting problem is rather limited. An early stopping approach even before obtaining its minimum and network pruning techniques are some of the indirect ways of controlling the over-fitting problem. SVM considers both training error and confidence interval (capacity of the system). The technique implemented in SVM controls these two items effectively. As a result, SVM possess a good generalization feature (better performance on unseen data).

3. ANNs can not avoid the risk of getting trapped in local minima while training, due to its formulation. SVM instead solves a quadratic programming problem, which has a unique optimal solution.
4. SVM uses the ε-insensitive loss function which filters noise of ε level. This provides SVM the robustness in dealing with real world noisy mixed time series. Other techniques do not have this feature.

2.3.4 Issues related to model parameters

Though SVM has various advantages as listed above, the parameter calibration remains an open issue. The parameters involved and selected by the user are:

1. Parameter C (Eq. (2.24)) controls the trade-off between the training error and the model complexity. Since SVM maps data into high dimensional feature space, C is sensitive to the model performance. Only a good choice of C can provide a good result.
2. Another parameter is ε from the ε-insensitive loss function. ε can be related to the noise of the training data. However the noise of the real world data is usually unknown.
3. Another parameter is σ, the width of the Gaussian kernel. It controls the complexity of the model. The smaller σ is, the more powerful SVM can approximate. The dimension of the feature space of Gaussian kernel is infinitely large. The results of SVM are implicitly provided from the feature space by using the kernel method.

These three major free parameters need to be calibrated before SVM can be utilized to its fullest. These parameters must be tuned simultaneously. It is a quite difficult problem for regression and there is, however, no good and efficient method available.

It is reported in SVM applications that tuning these parameters is largely a trial and error process (e.g. Vapnik et al., 1997; Muller et al., 1997; Dibike et al., 2001; Liong and Sivapragasam, 2002).

2.3.5 SVM for dynamics reconstruction of chaotic system

Müller et al. (1997) employed SVM for chaotic time series forecasting. The embedding parameter was found by the method of Liebert et al. (1991). They proposed to use SVM on artificial noise mixed Mackey-Glass and Santa Fe time series. The training data were categorized into several segments with each segment containing shorter records of about 300 patterns. A good performance was shown, better than MLP and RBF networks. They pointed out that the choice of the parameters (C, ε, σ) is suboptimal. They did not, however, explicitly indicate the difficulty of SVM in handling large training samples, which is a typical case for chaotic time series. It should be noted that there was still no efficient way to solve the quadratic programming problem of SVM for large data sets when Müller et al. published their papers in 1997.

Mattera and Haykin (1999) employed SVM on dynamics reconstruction of chaotic systems. Data used were the Lorenz time series. In their study, some empirical suggestions in choosing the parameters of SVM were given.

The ε-value shapes the actual loss function and affects the approximation error, training time, and complexity of the solution. The training time and complexity depend on the number of support vectors. The number of support vectors is a decreasing function of ε. Large ε-values can be utilized to reduce the training time and network complexity. Mattera and Haykin (1999) proposed to choose ε-value when the number of support vectors is about 50% of the whole training set.

C is proposed to be chosen as about equal to the range of the target variable (Mattera and Haykin, 1999). From Eq. (2.29), the following can be derived:

$$\begin{aligned} |y| &\le \left|\textstyle\sum_{i=1}^{N'}(\alpha_i - \alpha_i')K(\mathbf{x}_i,\mathbf{x})\right| \\ &\le \textstyle\sum_{i=1}^{N'}|\alpha_i - \alpha_i'||K(\mathbf{x}_i,\mathbf{x})| \le \textstyle\sum_{i=1}^{N'}C|K(\mathbf{x}_i,\mathbf{x})| \le C \times N' \end{aligned} \tag{2.33}$$

where N′ is the number of the support vectors. Denoting B = max | y |, set C ≥ B will satisfy any case with different number of support vectors. If C is very large compared to B, this will increase the linear coefficients (α - α′) and may give rise to numerical instability and cost unpleasantly long training time.

The difficulties in dealing with large chaotic time series were addressed in Mattera and Haykin (1999) as a ‘formidable’ problem. The QP solver used in the study was Minos 5.4 which implements quasi-Newton approximation and stores the Hessian matrix of the size of O(N^2) where N is the sample size of the training data. There was, however, still no efficient scheme to handle large data sets for regression problems at the time of their investigation in 1999.

2.3.6 Summary

SVM is a newly developed learning machine. It has been shown that SVM may provide better performance than common neural networks since SVM is based on the principle of structural risk minimization. During training, SVM finds the support vectors automatically. It therefore does not suffer from the structure determination problems. Since SVM has a unique optimal solution, it avoids the risk of getting trapped in local minima. SVM uses ε-insensitive loss function to filter out noises. This feature makes SVM more robust on real world’s noisy data.

Besides all the advantages described above, SVM has its own limitations. The parameters values selection remains an issue. As the training data size gets very large, quadratic programming is difficult to be solved by common techniques due to the tremendously large memory requirement and long computational time.

SVM has been employed to the dynamics reconstruction and forecasting; good performances have been demonstrated. Investigations of SVM applied for phase space reconstruction have, however, not been explored, partly because chaotic time series involve a large data set, and partly because SVM is a quite newly developed tool.

2.4 Conclusions

Some fundamental principles relating to Chaos and Chaotic theory were discussed. A detailed review on standard chaotic techniques and inverse approach for phase space reconstruction was conducted. It was demonstrated that in practice inverse approach is superior over the standard techniques. A brief discussion on approximation techniques involved in SVM is made. A detailed review on support vector machine together with its advantages and disadvantages was also provided.

This review pointed out the need: (1) to thoroughly explore the performance capability of SVM in chaotic time series; (2) for SVM to find an efficient scheme to deal with large data records; and (3) for SVM to find an efficient scheme to calibrate model parameters. The following chapters present schemes addressing the above issues.

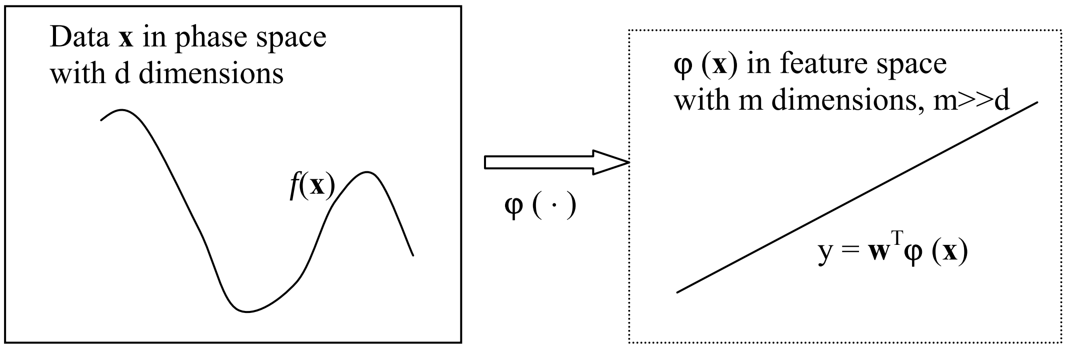


Figure 2.1 Illustration of data conversion from reconstructed phase space to feature space

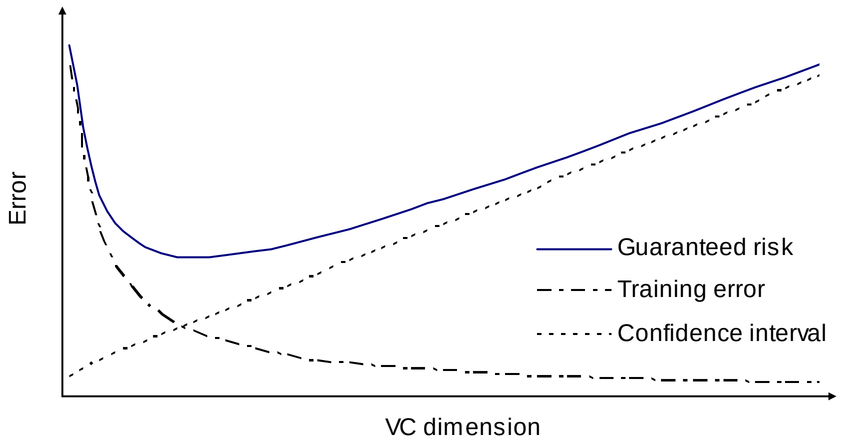
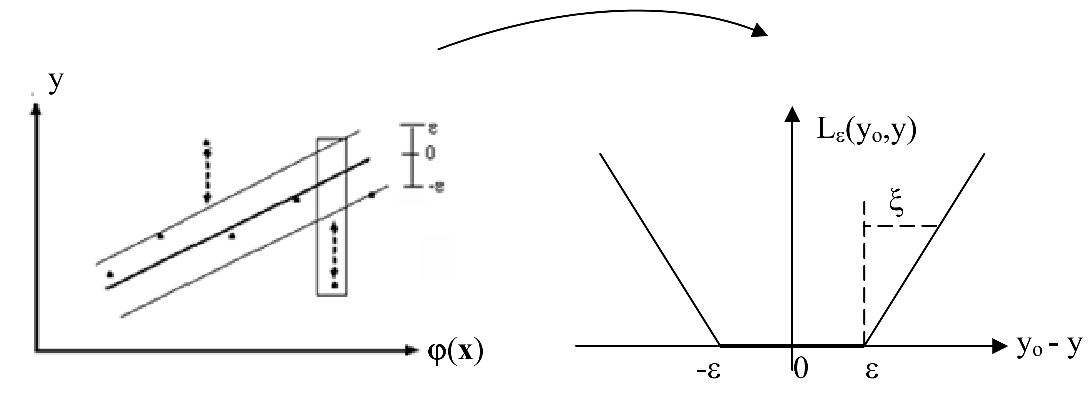
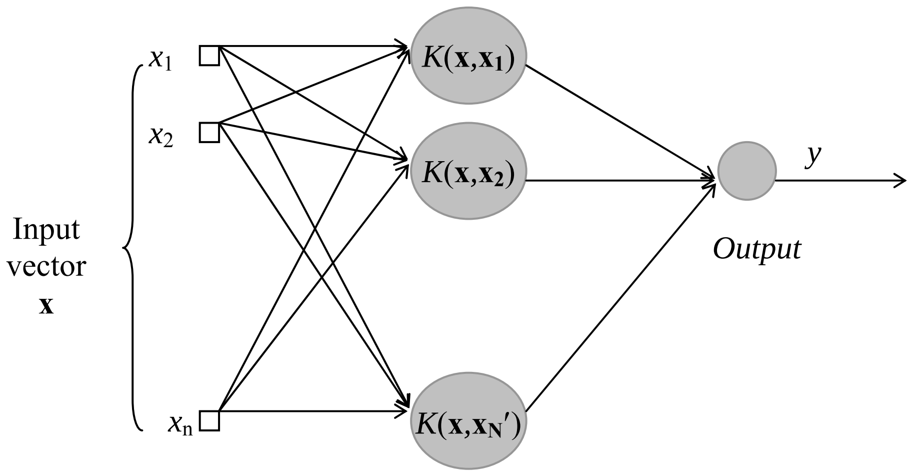


Figure 2.2 Illustration of structural risk minimization



(a) Data and best fit function in feature space (b) Penalty function

Figure 2.3 ε-insensitive loss function



x_1
x_2
x_n
Input vector **x**
$K(\mathbf{x},\mathbf{x_1})$
$K(\mathbf{x},\mathbf{x_2})$
$K(\mathbf{x},\mathbf{x_N}')$
y
Output

Figure 2.4 Architecture of Support Vector Machine (SVM)

CHAPTER 3
SVM FOR PHASE SPACE RECONSTRUCTION

3.1 Introduction

In this chapter SVM, used as an approximation engine, is applied first to reconstruct the phase space, with the least prediction error as the objective function, and then to do forecasting.

The analysis of chaotic time series is always associated with large scale data sets which pose serious difficulties to approximation techniques. This is the main reason why many studies favoured the local model due to its simplicity, although it is not so powerful as others. When SVM solves the quadratic programming, the Hessian matrix of size O(N^2) needs to be stored in memory all the time during the training task. As the training data size N is large, which is a common case for chaotic time series, the memory requirement is tremendously large and poses serious problem to common PCs. Hence an effective algorithm, which can deal with large scale data sets, is required in the analysis of chaotic time series.

In this chapter, the proposed application of SVM in the phase space reconstruction, and then for prediction, is first described. The implementation of special algorithms to deal with the inevitably large data set, required in the analysis of chaotic time series, then follows. This special algorithm is of utmost importance to make SVM of practical usage.

3.2 Proposed SVM for dynamics reconstruction

After the phase space reconstruction, the prediction problem is solved in normal Euclidean space. SVM, with its generalization capability, is very powerful to approximate nonlinear relationships. Using the lag vector as the input variable and the prediction variable as the target function, SVM can detect, on the basis of statistical learning, the underlying relationship effectively. In this section, the generalization capability of SVM will be explored.

3.2.1 Dynamics reconstruction with SVM

Dynamics reconstruction is operated in the reconstructed phase space. For a given time series, y_1, y_2, …, y_N, with a known time delay (τ), and embedding dimension (d), the lag vector input and the corresponding *l*-lead time output can be easily assembled and used for training, for example. Figure 3.1 shows a simple conversion example with τ = 1, d = 2, and *l* = 1.

After the embedding structure parameter set (d, τ) is determined by the standard approach such as AMI, and FNN, or by other techniques, for a given training data set, y_1, y_2,…, y_N, the input vector $\mathbf{y}_t$ and the output vector y_{t+l} are set up. The length of the data set for training is $N' = N - (d - 1) \times \tau - l$**.** The essential task in the prediction is to fit the relationship between the predicted variable and the lag vector:

$$y_{t+l}^{d} = f(\mathbf{y}_t) + v = f(y_t, y_{t-\tau}, \ldots, y_{t-(d-1)\tau}) \tag{3.1}$$

A widely used technique is the local model to fit this relationship using the local linear model as illustrated in Fig. 3.2. The local model finds a number of nearest neighbours among the training data and the prediction follows the pattern of these evolutions of the nearest neighbours.

This regression problem is carried out in the reconstructed phase space **y**. Instead of using a local model, the following model structure is proposed in this study: (1) the lag vector is used as the input; (2) a *l*-lead time prediction is the desired output; and (3) SVM is used for the regression problem, as given in Eq. (3.1). SVM is able to find a very good fit for complex nonlinear functions. SVM replaces the K nearest neighbour prediction engine, a local linear method. Figure 3.3 depicts a schematic diagram of how SVM is combined with chaos based techniques.

For a given parameter set of embedding structure of a time series, the training samples, $\{\mathbf{y}_i, y_{(t+l)oi}\}$ where i = 1, 2, …, N, provide the lag vector and the corresponding predicted vector to establish their relationship. Mathematically, the problem deals with an optimization of the following quadratic programming problem:

Maximize: $$Q(\alpha,\alpha') = \sum_{i=1}^{N} y_{(t+l)_{oi}}(\alpha_i - \alpha_i') - \varepsilon \sum_{i=1}^{N}(\alpha_i + \alpha_i') - \frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{N}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{y}_i,\mathbf{y}_j) \tag{3.2a}$$

Subject to: $$\sum_{i=1}^{N}(\alpha_i - \alpha_i') = 0 \tag{3.2b}$$

$$0 \le \alpha_i \le C, 0 \le \alpha_i' \le C, i = 1, 2, \ldots, N \tag{3.2c}$$

After finding the optimal Lagrange multipliers, the weights are then determined. The regression function *f* now becomes:

$$y_{t+l} = \sum_{i=1}^{N}(\alpha_i - \alpha_i')K(\mathbf{y},\mathbf{y}_i) \tag{3.3}$$

The flowchart of the forecasting task of chaotic system is given in Fig. 3.4.

3.2.2 Calibration of SVM parameters

SVM has several parameters which require calibration prior to its applications. There are some suggestions available in the literature. However, these suggestions are empirical and should be used as a guideline only. In this study, these parameters are calibrated with the least prediction error used as the objective function. The set of parameters, with which SVM provides best prediction on unseen data set, is chosen as the optimal set and will be used for the prediction task.

The data set is divided into three sets: (1) training set; (2) test set; and (3) validation set. Once the embedding structure parameter set is given, the lag vector can be constructed for the time series. The training set is used to select the SVM's support vectors and to solve the quadratic programming problem in obtaining the Lagrange coefficients, α and α′. The test set is used to select the optimal SVM parameters with the least prediction error used as the objective function. The root mean square error is used as criteria:

$$Error_{test} = \sqrt{\sum_{i=1}^{N'} (y(t+l)_{ip} - y(t+l)_{io})^2) / N'} \qquad (3.4)$$

The validation set is then used to validate the performance of the optimal SVM parameters determined from the use of training and test data sets.

The kernel used in this study is the Gaussian kernel. It has been reported in various publications that the Gaussian kernel provides better performance than the dot product kernel, polynomial kernel, and sigmoid kernel. Since the Gaussian parameter σ influences the complexity of SVM, a proper selection of σ is of utmost important.

There are three parameters in SVM: C, ε, and σ. The selection of these parameters has been the focus of many research works. C controls the trade-off between complexity of the machine and the empirical error. The higher the C value is, the more emphasis is placed on the empirical error. ε is the parameter in the ε-insensitive function and depends on the noise level of the original data set. The lower the ε value is, the lower level the noise is allowed, i.e., the higher the empirical error is.

σ^2 is the width of the radial basis function. The lower the σ value is, the more powerful the Kernel is, i.e., the more complicated the function it can approximate.

Figure 3.5 illustrates the selection procedure of the SVM parameters based on the prediction accuracy. Once the embedding structure parameter set is proposed, the best parameter set (C, ε, σ) will be chosen from the minimum test error of the test set.

3.3 Proposed SVM for phase space and dynamics reconstructions

Applying SVM first for phase space reconstruction and then for dynamics reconstruction is proposed in this study. The embedding structure parameter set is the corner stone in the phase space reconstruction. Figures 3.3 and 3.5 show how the choice of the embedding structure parameter set affects the prediction accuracy.

3.3.1 Motivations

Techniques commonly used to select the embedding structure parameters are AMI and FNN. In AMI, the proposed time lag is chosen when the first minimum of the mutual information arrives. AMI is related closely with entropy which characterizes the chaotic system. However, it is a known fact that there is no very strong theoretical ground in the choice of the exact value of time lag. It has been shown widely that time lag chosen from the proposed guideline does not necessarily provide good prediction for real time series. Moreover, for real time series the first minimum of the average mutual information is not very distinct. Figure 3.6 shows an example of a time series which displays broad band Fourier transform and low correlation dimension. The 'first' minimum of AMI may be taken at time lag of 12. The AMI value decreases very gradually already at time lag greater than 8 or so and the 'first' minimum of AMI value is really not significantly different from the values nearby. The choice therefore could be rather wide. Only after a ‘proper’ time lag has been selected, the selection of a proper embedding dimension with FNN can be conducted.

In the FNN method, the selection of the proper embedding dimension is completed once the false nearest neighbours have been eliminated. However, definition of false nearest neighbours is quite ambiguous. There is no strong theoretical ground to judge whether two points are false nearest neighbour. An empirical threshold value, 15, is used instead to decide if two points are false nearest neighbours. In principle, if the additional Euclidean distance with the increasing dimension exceeds the threshold value 15, then the ‘neighbours’ under consideration are identified as false nearest neighbours. A change in the threshold value will obviously affect the false nearest neighbours decision making.

Since the embedding parameters resulting from these commonly used embedding techniques do not guarantee ‘optimal’ prediction accuracy, the choice for the least prediction error as the objective function adopted in this study seems to be a reasonable proposal.

3.3.2 Proposed method

SVM will be used in this study to find the optimal embedding structure and SVM parameters with which the prediction error is the minimum. Figure 3.7 shows the basic idea of this novel method.

SVM functions as a regression engine in this study for chaotic time series analysis. Unlike the algorithm shown in Fig. 3.3, SVM is now used for both the phase space and dynamics reconstructions as illustrated in Fig. 3.7.

The data set is divided into three sets: (1) training set; (2) test set; and (3) validation set. A parameter set of embedding structure is first selected. Lag vectors are then constructed. The training set is used for SVM model to select the support vectors and to obtain the Lagrange coefficients, α and α′. The test set is used to select the optimal parameter set of embedding structure and the corresponding SVM model parameters based on the least performance error. The validation set is finally used to validate the performance of the optimal parameter set.

There are five parameters to be determined in this approach. They are: the time delay τ, the embedding dimension d, and the 3 parameters in SVM C, ε, σ. These five parameters have to be determined simultaneously. Figure 3.8 illustrates the procedures of the proposed approach.

It should be noted that an efficient technique able to deal with large training data set is highly essential for the success of this proposed method. Only when SVM powered with a fast and effective scheme for large data sets can the proposed SVM be competitive with or more superior over other already widely used techniques for chaotic time series analysis.

3.4 Handling of large data record with SVM

The dual problem of SVM deals with optimization of a quadratic objective function expressed in α_i, α_i'. A linear constraint accompanies the dual problem as given in Eq. (3.2). The objective function in Eq. (3.2a) is not in the standard form of qudratic programing and it can be converted to standard qudratic function as: $f(\mathbf{x}) = {}^1\!/_2\, \mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}$, where **H** is the Hessian matrix. With

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_1 \\ \dots \\ \alpha_N \end{bmatrix} \quad \boldsymbol{\alpha}' = \begin{bmatrix} \alpha_1' \\ \alpha_2' \\ \dots \\ \alpha_N' \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} k(\mathbf{y}_1,\mathbf{y}_1) & k(\mathbf{y}_1,\mathbf{y}_2) & \dots & k(\mathbf{y}_1,\mathbf{y}_N) \\ k(\mathbf{y}_2,\mathbf{y}_1) & k(\mathbf{y}_2,\mathbf{y}_2) & \dots & k(\mathbf{y}_2,\mathbf{y}_N) \\ \dots & \dots & \dots & \dots \\ k(\mathbf{y}_N,\mathbf{y}_1) & k(\mathbf{y}_N,\mathbf{y}_2) & \dots & k(\mathbf{y}_N,\mathbf{y}_N) \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_{o1} \\ y_{o2} \\ \dots \\ y_{oN} \end{bmatrix} \qquad (3.5)$$

and $\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\alpha} \\ -\boldsymbol{\alpha}' \end{bmatrix}$ $\quad \widetilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{bmatrix}$ $\quad \mathbf{b} = \begin{bmatrix} -\mathbf{Y} - \mathbf{1}\varepsilon \\ -\mathbf{Y} + \mathbf{1}\varepsilon \end{bmatrix}$ (3.6)

the dual problem, as given in Eq. (3.2), can be converted to the standard quadratic programming form as follows:

Minimize: $Q(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^T \widetilde{\mathbf{K}}\boldsymbol{\beta} - \boldsymbol{\beta}^T \mathbf{b}$ (3.7a)

Subject to: $\boldsymbol{\beta}^T \mathbf{1} = 0$ (3.7b)

$0 \le \delta_i \beta_i \le C$, $i = 1, 2, \ldots, 2N$ (3.7c)

$\delta_i = 1$ for $1 \le i \le N$ and $\delta_i = -1$ for $N+1 \le i \le 2N$. SVM deals with a quadratic programming with one linear constraint and bound constraints. Even though this type of optimisation problem is well understood and algorithms are well developed, a serious obstacle is faced when it deals with a large training data set. The Hessian matrix, Eq. (3.8), becomes tremendously large with increasing training sample size:

$$\widetilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} \end{bmatrix} = \begin{bmatrix} k(\mathbf{y}_1,\mathbf{y}_1) & \ldots & k(\mathbf{y}_1,\mathbf{y}_N) & k(\mathbf{y}_1,\mathbf{y}_1) & \ldots & k(\mathbf{y}_1,\mathbf{y}_N) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ k(\mathbf{y}_N,\mathbf{y}_1) & \ldots & k(\mathbf{y}_N,\mathbf{y}_N) & k(\mathbf{y}_N,\mathbf{y}_1) & \ldots & k(\mathbf{y}_N,\mathbf{y}_N) \\ k(\mathbf{y}_1,\mathbf{y}_1) & \ldots & k(\mathbf{y}_1,\mathbf{y}_N) & k(\mathbf{y}_1,\mathbf{y}_1) & \ldots & k(\mathbf{y}_1,\mathbf{y}_N) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ k(\mathbf{y}_N,\mathbf{y}_1) & \ldots & k(\mathbf{y}_N,\mathbf{y}_N) & k(\mathbf{y}_N,\mathbf{y}_1) & \ldots & k(\mathbf{y}_N,\mathbf{y}_N) \end{bmatrix} \quad (3.8)$$

For instance, a 20 years daily flow time series has about 7,300 records. The Hessian matrix has the size of square of 2 times the sample size, i.e. 213,160,000. If each element of the Hessian matrix is stored as an 8-byte double precision number, the total memory capacity required is 1,705 Megabyte. Common PCs have a RAM of size 256 Megabyte. Since the Hessian matrix is required to be stored, this requirement poses a serious problem for SVM.

3.4.1 Decomposition method

Most recently a decomposition method was developed to overcome the above mentioned problem. This allows SVM to deal with large data record problem. For classification problems, Platt (1999) developed sequential minimal optimisation (SMO) and Joachims (1999) developed SVM^{light}. For regression problem, Collobert and Bengio (2001) successfully implemented the decomposition method in SVM^{light}. Its ability to handle large data sets was demonstrated on the regression problems as robot arm moving as a function of 32 variables like joint position, twist angle etc. with 6192 training examples, yearly average sunspot as a function of 12 previous yearly averages with 40,000 training samples. In this study, the scheme is introduced to the chaotic time series analysis.

3.4.1.1 Introduction

The basic idea of the decomposition method is to decompose the quadratic programming problem into a series of small quadratic programming problem of only 2 selected variables while the remaining variables are fixed. The memory requirement is then significantly decreased into O(m×N), where m is a small value integer. Since a quadratic programming with 2 variables can be solved analytically, the whole algorithm becomes very efficient. The basic algorithm is as follows:

[1] Set an initial value β^0 for all $\boldsymbol{\beta}$;

[2] Select 2 working variables among 2N variables, e.g. β_1, β_2, among $\boldsymbol{\beta}$;

[3] Solve the quadratic programming having only 2 variables analytically. $Q(\boldsymbol{\beta}^{k+1}) < Q(\boldsymbol{\beta}^k)$ is guaranteed;

[4] Check the optimal conditions. If the KKT conditions are met, the optimum has been achieved; otherwise, go to step [2] and repeat the remaining steps.

The decomposition method splits the variables into a fixed set F and a working set S. Denoting:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}s \\ \boldsymbol{\beta}_F \end{bmatrix} \quad \tilde{\mathbf{K}} = \begin{bmatrix} \tilde{\mathbf{K}}ss & \tilde{\mathbf{K}}_{SF} \\ \tilde{\mathbf{K}}_{FS} & \tilde{\mathbf{K}}_{FF} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}s \\ \mathbf{b}_F \end{bmatrix} \tag{3.9}$$

$\boldsymbol{\beta}s$ contains 2 variables, e.g. β_1, β_2, which are selected as the working variables among $\boldsymbol{\beta}$. The objective function, given in Eq. (3.7a), becomes a minimization problem of:

$$\begin{aligned}
Q(\boldsymbol{\beta}) &= \frac{1}{2}\boldsymbol{\beta}^T\tilde{\mathbf{K}}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{b} = \frac{1}{2}\begin{bmatrix}\boldsymbol{\beta}_S^T & \boldsymbol{\beta}_F^T\end{bmatrix} \times \begin{bmatrix} \tilde{\mathbf{K}}ss & \tilde{\mathbf{K}}_{SF} \\ \tilde{\mathbf{K}}_{FS} & \tilde{\mathbf{K}}_{FF} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{\beta}s \\ \boldsymbol{\beta}_F \end{bmatrix} - \begin{bmatrix}\boldsymbol{\beta}_S^T & \boldsymbol{\beta}_F^T\end{bmatrix} \times \begin{bmatrix} \mathbf{b}s \\ \mathbf{b}_F \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix}\boldsymbol{\beta}_S^T\tilde{\mathbf{K}}ss + \boldsymbol{\beta}_F^T\tilde{\mathbf{K}}_{FS} & \boldsymbol{\beta}_S^T\tilde{\mathbf{K}}_{SF} + \boldsymbol{\beta}_F^T\tilde{\mathbf{K}}_{FF}\end{bmatrix} \times \begin{bmatrix} \boldsymbol{\beta}s \\ \boldsymbol{\beta}_F \end{bmatrix} - \left(\boldsymbol{\beta}_S^T\mathbf{b}s + \boldsymbol{\beta}_F^T\mathbf{b}_F\right) \\
&= \frac{1}{2}\left(\boldsymbol{\beta}_S^T\tilde{\mathbf{K}}ss\boldsymbol{\beta}s + \boldsymbol{\beta}_F^T\tilde{\mathbf{K}}_{FS}\boldsymbol{\beta}s + \boldsymbol{\beta}_S^T\tilde{\mathbf{K}}_{SF}\boldsymbol{\beta}_F + \boldsymbol{\beta}_F^T\tilde{\mathbf{K}}_{FF}\boldsymbol{\beta}_F\right) - \left(\boldsymbol{\beta}_S^T\mathbf{b}s + \boldsymbol{\beta}_F^T\mathbf{b}_F\right) \\
&= \frac{1}{2}\boldsymbol{\beta}_S^T\tilde{\mathbf{K}}ss\boldsymbol{\beta}s - \boldsymbol{\beta}_S^T(\mathbf{b}s - \tilde{\mathbf{K}}_{SF}\boldsymbol{\beta}_F) + \frac{1}{2}\boldsymbol{\beta}_F^T\tilde{\mathbf{K}}_{FF}\boldsymbol{\beta}_F - \boldsymbol{\beta}_F^T\mathbf{b}_F
\end{aligned} \tag{3.10}$$

Denoting $\mathbf{h} = \mathbf{b}s - \tilde{\mathbf{K}}_{SF}\boldsymbol{\beta}_F$, it is equivalent to the following standard quadratic programming form:

Minimize: $Q(\boldsymbol{\beta}s) = \frac{1}{2}\boldsymbol{\beta}_s^T\tilde{\mathbf{K}}ss\boldsymbol{\beta}s - \boldsymbol{\beta}_S^T\mathbf{h}$ (3.11a)

Subject to: $\boldsymbol{\beta}_S^T\mathbf{1} = -\boldsymbol{\beta}_F^T\mathbf{1}$ (3.11b)

$0 \le \delta_i\beta_i \le C$, $i = 1, 2, \ldots, 2N$ (3.11c)

Only $\tilde{\mathbf{K}}_{SF}$ and $\tilde{\mathbf{K}}_{SS}$ are required to be stored in the memory, i.e. 2 rows, corresponding to the 2 selected working variables, of the $\tilde{\mathbf{K}}$ matrix, as shown in Fig. 3.9 (b). The rest rows, $\tilde{\mathbf{K}}_{FF}$, are not required. The memory requirement is decreased from the square of 2 times the sample size, $4N^2$, to 4 times the sample size, 4N, as shown in Fig. 3.9.

3.4.1.2 Brief description of technique

The decomposition technique used here adopts 2 working variables and the selection of the 2 working variables is based on the feasible direction method. The stopping criteria are the Karush-Khun-Tucker (KKT) conditions.

(1) Two working variables

βs contains only 2 variables: β_1, β_2. The optimization problem can be expressed as follows. With

$$\boldsymbol{\beta}s = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad \tilde{\mathbf{K}}ss = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \quad \zeta = -\boldsymbol{\beta}_F^T \mathbf{1} \tag{3.12}$$

Equation (3.11) becomes:

Minimize: $Q(\beta_1, \beta_2) = \frac{1}{2}[\beta_1 \quad \beta_2] \times \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \times \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} - [\beta_1 \quad \beta_2] \times \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$ (3.13a)

Subject to: $\beta_1 + \beta_2 = \zeta$ (3.13b)

$0 \le \delta_1\beta_1, \delta_2\beta_2 \le C$ (3.13c)

With $\beta_2 = \zeta - \beta_1$, the above objective function becomes:

Minimize: $Q(\beta_1) = \frac{1}{2}(k_{11} - 2k_{12} + k_{22})\beta_1^2 + [(k_{12} - k_{22})\zeta - h_1 - h_2]\beta_1$ (3.14a)

This is a simple quadratic program with one variable as the standard form:

$f(x) = {}^1/_2\, ax^2 + bx.$ (3.14b)

$a = k_{11} - 2k_{12} + k_{22} > 0$ always holds for Gaussian kernel. The function has a unique minimum when $\beta_1 = a/b$. The solution depends on the bound constraints of β_1. If a/b is within the bound constraints, a/b is the solution. Otherwise, the solution is one of the boundary points as shown in Fig. 3.10. Once one of the working variable, β_1, is solved, the other working variable, β_2, can be easily obtained from $\beta_2 = \zeta - \beta_1$.

(2) Selection of 2 working variables

There are a number of choices to select 2 working variables among 2*N* variables. The total number of choices is:

$$\frac{1}{2}C_{2N}^{2} = \frac{1}{2}\frac{(2N)!}{(2N-2)!} \tag{3.15}$$

Choosing a good working set is highly essential to ensure a rapid convergence. Thus, an efficient and effective selection method is the key to the minimization of the objective function Q($\boldsymbol{\beta}$). The strategy is based on Zoutendijk's feasible direction method (FDM) for constrained optimisation problem (Zoutendijk, 1970). Steepest feasible descent strategy is used to choose a good pair of working variables and guarantees that the variables selected have the largest potential to minimize the objective function.

Figure 3.11(a) shows how the optimisation of the decomposition method progresses. $\boldsymbol{\beta}$ is varying within the feasible region as $\boldsymbol{\beta}$ approaches to the optimum. At an iteration *k*, for example, $\boldsymbol{\beta}^k = (\beta^k_1, \beta^k_2, \beta^k_3, \ldots, \beta^k_{2N})$ and only β^k_1 and β^k_2 are chosen as the working variables. Thus, only β^k_1 and β^k_2 become β^{k+1}_1 and β^{k+1}_2 at the iteration (k+1) while the rest $\beta^k_3, \beta^k_4, \ldots, \beta^k_{2N}$ remain unchanged, i.e. $\boldsymbol{\beta}^{k+1} = (\beta^{k+1}_1, \beta^{k+1}_2, \beta^k_3, \ldots, \beta^k_{2N})$. Denoting **d** as the difference between $\boldsymbol{\beta}^{k+1}$ and $\boldsymbol{\beta}^k$:

$$\mathbf{d} = \boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^{k} \tag{3.16}$$

d has only 2 nonzero components, i.e. **d** = (d_1, d_2, 0, …, 0). Since the linear constraint $\beta^T\mathbf{1} = 0$ must hold, $(\beta^{k+1} - \beta^k)^T\mathbf{1} = 0$ is true, i.e. $\mathbf{d}^T 1 = 0$ or $d_1 + d_2 = 0$. When the problem is projected on $\beta_1\beta_2$ space, the feasible region is a line of tangent equal to –1 ($\beta_1+\beta_2 =$ constant, as in Eq. (3.12b)) and there are only 2 possible directions the solution points can move as illustrated in Fig. 3.11 (b).

To choose a good set of working variables, steepest feasible descent strategy is employed. The less the dot product of the gradient $\nabla Q(\beta)$ and **d** is, the closer **d** is to the negative gradient; this means that the working variables will reduce further the objective function $Q(\beta)$. For instance, direction 2 in Fig. 3.11 (b) will be chosen among the four possible directions. A good working set can be found by solving:

$$\text{Minimize: } \nabla Q(\boldsymbol{\beta}^k)\mathbf{d} / \|\mathbf{d}\| \tag{3.17}$$

Since **d** has only two nonzero components and $d_1 + d_2 = 0$, the above problem is reduced to:

$$\text{Minimize: } (Q'_{\beta_1{}^k} - Q'_{\beta_2{}^k}) / \sqrt{2} \tag{3.18}$$

This translates to a problem finding the minimum ($Q'_{\beta 1}$ - $Q'_{\beta 2}$). Therefore, the two working variables should be as such that one variable (β_1) has the smallest first order derivative $Q'_{\beta 1,}$ among the total 2N variables, while the other variable (β_2) has largest first order derivative $Q'_{\beta 2}$.

(3) Checking of KKT condition

Since SVM solves a quadratic programming, there is a unique optimal solution to this quadratic programming. The Karush-Khun-Tucker (KKT) condition is the necessary and sufficient condition for the optimal solution. Thus, the KKT condition is used for checking whether the algorithm has achieved the optimal solution.

For optimization problem stated in Eq. (3.2), KKT condition holds at the optimal point:

$$Q'(\boldsymbol{\alpha},\boldsymbol{\alpha}') + \lambda^{eq}\begin{bmatrix} \mathbf{1} \\ -\mathbf{1} \end{bmatrix} - \boldsymbol{\lambda}^{low} + \boldsymbol{\lambda}^{up} = \mathbf{0} \qquad (3.19a)$$

$$\lambda_i^{low}\alpha_i^{(')} = 0; \qquad (3.19b)$$

$$\lambda_i^{up}(\alpha_i^{(')} - C) = 0 \qquad (3.19c)$$

Equations (3.19b) and (3.19c) imply that:

$$\alpha_i^{(')} = C, \Rightarrow \lambda_i^{low} = 0, \lambda_i^{up} > 0; \qquad (3.20a)$$

$$\alpha_i^{(')} = 0, \Rightarrow \lambda_i^{up} = 0, \lambda_i^{low} > 0; \qquad (3.20b)$$

$$0 < \alpha_i^{(')} < C, \Rightarrow \lambda_i^{up} = \lambda_i^{low} = 0 \qquad (3.20c)$$

Therefore λ^{eq} can be estimated by those $0 < \alpha_i^{(')} < C$, i.e. $\lambda_i^{up}=\lambda_i^{low} = 0$. Noting:

A=$\{i, 0 \le \alpha_i \le C\}$,   B=$\{i, 0 \le \alpha'_i \le C\}$

Applying $\lambda_i^{up}=\lambda_i^{low}=0$, as in Eq. (3.20c), to Eq. (3.19a), λ^{eq} may then be estimated by:

$$\hat{\lambda}^{eq} = \frac{1}{|A \cup B|}\left(\sum_{i\in B} Q'_{i+l}(\boldsymbol{\alpha},\boldsymbol{\alpha}') - \sum_{i\in A} Q'_i(\boldsymbol{\alpha},\boldsymbol{\alpha}') \right) \qquad (3.21)$$

When $\lambda_i^{up} > 0$ as in Eq(3.20a), $\lambda_i^{low} > 0$ as in Eq(3.20b), and $\lambda_i^{up/low} = 0$ as in Eq.(3.20c), are verified, i.e.:

$$\delta_i\beta_i = C, \quad \text{if } \lambda_i^{up} = -(Q'(\boldsymbol{\alpha},\boldsymbol{\alpha}') + \lambda^{eq}\delta_i) \ge -\varepsilon_{end} \qquad (3.22a)$$

$$\beta_i = 0, \quad \text{if } \lambda_i^{low} = (Q'(\boldsymbol{\alpha},\boldsymbol{\alpha}') + \lambda^{eq}\delta_i) \ge -\varepsilon_{end} \qquad (3.22b)$$

$$0 < \delta_i\beta_i < C, \text{ if } \lambda_i^{up/low} = \left|Q'(\boldsymbol{\alpha},\boldsymbol{\alpha}') + \lambda^{eq}\delta_i\right| < \varepsilon_{end} \qquad (3.22c)$$

the optimal solution for the quadratic programming of Eq. (3.2) of the dual problem in SVM is achieved. When $\varepsilon_{end} = 0$, the above conditions are as strictly the same as Eq. (3.20). For numerical calculation, it is hardly to achieve $\varepsilon_{end} = 0$. Normally $\varepsilon_{end} = 0.01$ is set and setting a higher accuracy level will lead to a considerable longer training time.

3.4.1.3 Implementation

The decomposition method illustrated here is highly effective and efficient for large scale training set. Two key strategies employed in the algorithm: 2 working variables and the steepest feasible direction to select the 2 working variables. The algorithm converts the problem into a series of quadratic programming problems each having only two variables and one linear constraint.

Another technique attempts to speed up the training is the shrinking technique. For $\alpha=0$, λ^{low} can be estimated; if $\lambda^{low} > 0$ strictly holds for a long time, then it may be removed from the problem, as shown in Fig. 3.12. The algorithm converges when working size is equal to 2 and without shrinking. Shrinking is a heuristic and it will speed up the algorithm, but no convergence proof is available.

As described above, SVM equipped with a decomposition method could easily deal with large data record requirement. The software used in this study is SVMTorch written in C programming and running on Linux platform. There are three remaining parameters as described before, C, ε, σ, which will be calibrated together with the embedding structure parameters (τ, d).

3.4.2 Linear ridge regression in approximated feature space

The solid convergence of decomposition method to deal with large data sets has been demonstrated and proven (Collobert and Bengio, 2000). However, the iterative scheme employed in the decomposition method may not be efficient and hence yield a slow convergence. For example, different time series of the same length may have very different training times.

Most recently, Suykens et al. (2002) demonstrated that dual problem with kernel trick is suitable for large dimensional input space, while the feature space problem is more suitable for large data sets. They demonstrated the scheme on the Sinc function, a benchmark SVM regression problem, with 20,000 examples. In this study, this scheme is introduced to the chaotic time series analysis to deal with its large data sets problem. The scheme performs linear ridge regression between the target function and the features directly. It is known that the Gaussian kernel has infinitive dimension. The scheme offers a meaningful and effective approach to approximate its appropriate features. It approximates the eigenfunctions and eigenvalues, according to Mercer's theorem, with the use of a set of sample from the input space.

3.4.2.1 Brief description of technique

(1) Eigenfunctions and Eigenvalues approximation

Recently, Williams and Seeger (2000, 2001) pointed out that for learning task there is a probability density function in input space, $p(\mathbf{x})$, which should be included in the integral equation in Mercer's theorem:

$$\lambda\phi(\mathbf{x}') = \int K(\mathbf{x},\mathbf{x}')p(\mathbf{x})\phi(\mathbf{x})d\mathbf{x} \tag{3.23}$$

The eigenfunctions ϕ_j are orthonormal with respect to $p(\mathbf{x})$, i.e.,

$$\int \phi_i(\mathbf{x})p(\mathbf{x})\phi_j(\mathbf{x})d\mathbf{x} = \delta_{ij} \tag{3.24}$$

$\delta_{ij} = 1$ when $i = j$ and $\delta_{ij}=0$ when $i \neq j$. The kernel function used in SVM must be positive definite, i.e. for all functions $f(\mathbf{x}) \in \Lambda^2$, ($\int f^2(\mathbf{x})d\mathbf{x} < \infty$),

$$\int\int K(\mathbf{x},\mathbf{x}')f(\mathbf{x})f(\mathbf{x}')d\mathbf{x}d\mathbf{x}' < \infty . \tag{3.25}$$

K(**x**, **x′**) can be expanded into a uniformly convergent series with eigenvalue λ_j and eigenfunction ϕ_j. The expansion is as follows:

$$K(\mathbf{x},\mathbf{x}') = \sum_{j=1}^{N_F} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{x}') \tag{3.26}$$

where $N_F \le \infty$ is the number of positive eigenvalues. The following relationship holds between the features and the eigenvalues and eigenfunctions:

$$\varphi_j(\mathbf{x}) = \sqrt{\lambda_j}\,\phi_j(\mathbf{x}) \tag{3.27}$$

The approximation of the features, φ_j, can be obtained by approximating the eigenvalue λ_j and the eigenfunction ϕ_j. Given a random sample $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ from $p(\mathbf{x})$, the empirical estimation holds for $p(\mathbf{x})$,

$$p(\mathbf{x}) \cong 1/q \tag{3.28}$$

Introducing Eq. (3.28) into Eq. (3.24) and the integral in Eq. (3.23) yields:

$$\int \phi_i(\mathbf{x}) p(\mathbf{x}) \phi_j(\mathbf{x}) d\mathbf{x} \cong \tfrac{1}{q} \textstyle\sum_{i=1}^{q} \phi_i(\mathbf{x}) \phi_j(\mathbf{x}) = \delta_{ij} \tag{3.29a}$$

$$\int K(\mathbf{x},\mathbf{y}) p(\mathbf{x}) \phi(\mathbf{y}) d\mathbf{y} \cong \tfrac{1}{q} \sum_{k=1}^{q} K(\mathbf{x},\mathbf{x}_k) \phi_i(\mathbf{x}_k) \tag{3.29b}$$

Equation (3.23) now becomes:

$$\frac{1}{q} \sum_{k=1}^{q} K(\mathbf{x},\mathbf{x}_k) \phi_i(\mathbf{x}_k) \cong \lambda_i \phi_i(\mathbf{x}) \tag{3.30}$$

Introducing the sample $\mathbf{x}_j$ for $\mathbf{x}$ into Eq. (3.30) results in:

$$\frac{1}{q} \sum_{k=1}^{q} K(\mathbf{x}_j,\mathbf{x}_k) \phi_i(\mathbf{x}_k) \cong \lambda_i \phi_i(\mathbf{x}_j) \tag{3.31}$$

If λ_i and ϕ_i ($\mathbf{x}_k$) can be estimated as node points, ϕ_i ($\mathbf{x}$) can then be interpolated as:

$$\phi_i(\mathbf{x}) \cong \frac{1}{\lambda_i q} \sum_{k=1}^{q} K(\mathbf{x}, \mathbf{x}_k)\phi_i(\mathbf{x}_k) \tag{3.32}$$

The eigenvalues and eigenfunctions can be estimated as related to the eigen decomposition of the kernel matrix of the sample points. The kernel matrix of these *q* sample, $\mathbf{K}^{(q)}$, can be written as:

$$\mathbf{K}^{(q)} = \begin{bmatrix} K_{11} & K_{12} & \dots & K_{1q} \\ K_{21} & K_{22} & \dots & K_{2q} \\ \dots & \dots & \dots & \dots \\ K_{q1} & K_{q2} & \dots & K_{qq} \end{bmatrix} \tag{3.33}$$

The eigen decomposition of matrix $\mathbf{K}^{(q)}$ is expressed as follows:

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1q} \\ K_{21} & K_{22} & \dots & K_{2q} \\ \dots & \dots & \dots & \dots \\ K_{q1} & K_{q2} & \dots & K_{qq} \end{bmatrix} \times \begin{bmatrix} U_1^1 & U_1^2 & \dots & U_1^q \\ U_2^1 & U_2^2 & \dots & U_2^q \\ \dots & \dots & \dots & \dots \\ U_q^1 & U_q^2 & \dots & U_q^q \end{bmatrix}$$

$$= \begin{bmatrix} U_1^1 & U_1^2 & \dots & U_1^q \\ U_2^1 & U_2^2 & \dots & U_2^q \\ \dots & \dots & \dots & \dots \\ U_q^1 & U_q^2 & \dots & U_q^q \end{bmatrix} \times \begin{bmatrix} \lambda_1^{(q)} & & & \\ & \lambda_2^{(q)} & & \\ & & \dots & \\ & & & \lambda_q^{(q)} \end{bmatrix}$$

or $$\mathbf{K}^{(q)}\mathbf{U}^{(q)} = \mathbf{U}^{(q)}\mathbf{\Lambda}^{(q)} \tag{3.34}$$

where $\mathbf{U}^i$ is the eigenvector and $\lambda_i^{(q)}$ is the eigenvalue of matrix $\mathbf{K}^{(q)}$. Equation (3.34) and Eq. (3.31) match with each other. The following approximation is made:

$$\lambda_i \cong \frac{1}{q}\lambda_i^{(q)} \tag{3.35a}$$

$$\phi_i(\mathbf{x}_j) \cong \sqrt{q}U_j^i = \sqrt{q}U_{j,i}^{(q)} \tag{3.35b}$$

In Eq. (3.35b), a term $\sqrt{q}$ appears before U_{ji}, due to the need for ϕ_i to meet the same requirement as that in Eq. (3.29a). Introducing Eq. (3.35) in Eq. (3.32) and then in Eq. (3.27), the estimated feature space can be computed through:

$$\varphi_i(\mathbf{x}) = \frac{1}{\sqrt{\lambda_i^{(q)}}} \sum_{k=1}^{q} K(\mathbf{x}, \mathbf{x}_k) U_{k,i}^{(q)} \tag{3.36}$$

(2) Quadratic Renyi entropy for selection of the subset

The random sample $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ should be a subset of the entire sample. The memory requirement to store the features of all the training records is N×q×8byte. For example, for N = 7300 and q = 500, the memory requirement is 29MB. Like any good sampled points, the chosen q points should be a good representation of the whole sample points. The selection of these q points from the training data set can be made based on the quadratic Renyi entropy defined as:

$$H_R = -\log \int p(\mathbf{x})^2 d\mathbf{x} \tag{3.37}$$

For Gaussian kernel, $\int p(\mathbf{x})^2 d\mathbf{x}$ can be estimated (Girolami, 2002) by the sample points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ as:

$$\int p(\mathbf{x})^2 d\mathbf{x} = \frac{1}{q^2} \sum_{i=1}^{q} \sum_{j=1}^{q} K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.38}$$

Thus, the quadratic Renyi entropy can be estimated as:

$$H_R = -\log\left(\frac{1}{q^2} \sum_{i=1}^{q} \sum_{j=1}^{q} K(\mathbf{x}_i, \mathbf{x}_j) \right) \tag{3.39}$$

The quadratic Renyi entropy can be used as the criterion to choose a good set of $S=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$, which provides larger entropy, H_R. A subset with larger value of Renyi entropy is more scattered and therefore it represents the whole data set better.

$\{-\log \int p(\mathbf{x})^2 d\mathbf{x}\}$ decreases as $\{\int p(\mathbf{x})^2 d\mathbf{x}\}$ increases as shown in Fig. 3.13(a). A subset with large value of entropy means that this subset has a relatively low value of $\{\int p(\mathbf{x})^2 d\mathbf{x}\}$. From Eq. (3.38), this implies that the average of the kernel matrix is smaller. For instance when q=2, i.e. only two points are selected. Equation (3.38) becomes:

$$\frac{1}{2^2}\left(K_{11} + K_{22} + 2K_{12}\right) = 0.5 + 0.5K_{12} \tag{3.40}$$

K_{12} is equal to 0.2 and 0.6 respectively for cases A and B in Fig.3.13 (a). Figure 3.13 (b) shows the corresponding situation of cases A and B. The higher the entropy is, the larger the distance of the selected points. For situations in which q > 2, the same reason is applicable as that given for the case when q=2. Selecting the subset with largest entropy means selecting points that are most scattered and therefore should represent the whole data set best.

(3) Ridge linear regression

Ridge regression reduces the effective number of parameters. This results in a less sensitive model and hence a less overfitted model. For a given sample set $\{\mathbf{x}_i, y_{o(i+l)}\}$, where i=1, 2,…, N, the ridge regression, between the forecasting variable $y_{o(i+l)}$ and the approximated features $\{\varphi_j(\mathbf{x}_i)\}$,where j = 1, 2, …, q, minimizes the following cost function:

$$\text{Minimize } L(\mathbf{w}) = \sum_{j=1}^{N}(y_{o(j+l)} - \sum_{i=1}^{q} w_i\varphi_i(\mathbf{x}_j))^2 + C'\sum_{i=1}^{q} w_i^2 \tag{3.41}$$

The solution occurs when their derivatives with respect to w_i are equal to zero, i.e.,

$$\frac{\partial L}{\partial w_i} = 2\sum_{j=1}^{N}(y_{o(j+l)} - \sum_{i=1}^{q} w_i\varphi_i(\mathbf{x}_j))(-\varphi_i(\mathbf{x}_j)) + 2C'w_i = 0 \tag{3.42}$$

$$\sum_{j=1}^{N}(\sum_{i=1}^{q} w_i\varphi_i(\mathbf{x}_j))\varphi_i(\mathbf{x}_j) + C'w_i = \sum_{j=1}^{N} y_{o(j+l)}\varphi_i(\mathbf{x}_j),\ i=1, 2, \ldots, q \tag{3.43}$$

Denoting

$$\mathbf{H} = \begin{bmatrix} \varphi_1(\mathbf{x}_1) & \varphi_2(\mathbf{x}_1) & \dots & \varphi_q(\mathbf{x}_1) \\ \varphi_1(\mathbf{x}_2) & \varphi_2(\mathbf{x}_2) & \dots & \varphi_q(\mathbf{x}_2) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \varphi_1(\mathbf{x}_N) & \varphi_2(\mathbf{x}_N) & \dots & \varphi_q(\mathbf{x}_N) \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ .. \\ w_q \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_{o(1+l)} \\ y_{o(2+l)} \\ \dots \\ \dots \\ y_{o(N+l)} \end{bmatrix} \tag{3.44}$$

Equation (3.43) can now be expressed as:

$$\mathbf{H}^T\mathbf{H}\mathbf{w} + C'\mathbf{w} = \mathbf{H}^T\mathbf{y} \tag{3.45}$$

with solution as :

$$\mathbf{w} = (\mathbf{H}^T\mathbf{H} + C'\mathbf{I})^{-1}\mathbf{H}^T\mathbf{y} \tag{3.46}$$

The inverse may not exist in some matrices. In order to guarantee the scheme to be stable and reliable, the pseudo-inverse of ($\mathbf{H}^T\mathbf{H}$+C′$\mathbf{I}$) is used instead of the inverse.

3.4.2.3 Implementation

There are three major steps in SVM equipped with the linear regression in the approximated feature space, as illustrated in Fig. 3.14 as well,

Step I Select a good data subset, with the largest entropy, from the whole training set;

Step II Approximate the features by using the eigenvalues and engenfunctions of the kernel matrix from the selected subset;

Step III Apply ridge linear regression to fit the relationship of the target variable and the features.

In the selection of a good subset with largest entropy (Step I), the algorithm is as follows:

(1) Choose an initial working set of q points randomly, and calculate its entropy, H_R;

(2) Randomly select a sample $\mathbf{x}^*$ from the working set and also randomly select a sample $\mathbf{x}^{t*}$ from the whole training set, however, excluding the working set. Calculate the entropy H_R' of these new q points, with $\mathbf{x}^{t*}$ replacing $\mathbf{x}^*$. If the entropy increases, i.e., $H_R' > H_R$, replace the working set $\mathbf{x}^*$ by $\mathbf{x}^{t*}$;

(3) Stop when the difference between H_R' and H_R is small after a sufficiently large number of iterations, e.g. 100 iterations, otherwise repeat step 2.

In approximating the appropriate features (Step II), the required measures are:

(1) Eigen-decomposition of the matrix $\mathbf{K}_{q\times q}$ to determine both the eigenvalues and eigenvectors; and

(2) Features estimation of the whole sample points resulting from Eq. (3.36).

The accuracy of the eigenvalue and eigenvector of matrix $\mathbf{K}_{q\times q}$ highly influences the accuracy of the features estimation and therefore influences the final estimation. To avoid numerical instabilities of the eigendecomposition, it is a common practice to use a jitter factor α, e.g. $\alpha = 2$, which is a small positive constant. Eigen-decomposition is applied on $\{\mathbf{K}_{q\times q} + \alpha\mathbf{I}\}$ instead of on $\mathbf{K}_{q\times q}$ directly. The eigenvectors of $\mathbf{K}_{q\times q}$ are the same as that of $\{\mathbf{K}_{q\times q}+\alpha\mathbf{I}\}$. The relationship between eigenvalues of $\mathbf{K}_{q\times q}$, λ_j , and eigenvalues of $\{\mathbf{K}_{q\times q}+\alpha\mathbf{I}\}$, λ_j', is as follows:

$$\lambda_j = (\lambda_j' - \alpha)/q \tag{3.47}$$

Those λs with very small value, i.e. $\lambda<1.0$ e-10 including zeros, are neglected in calculating the features with the use of Eq. (3.36) to avoid any numerical instabilities.

In ridge linear regression (Step III), the objective is to obtain the value of the $\mathbf{w}$ by the matrices calculation of the right hand side of Eq. (3.46). The essential part of this step is to use pseudo-inverse of $(\mathbf{H}^T\mathbf{H}+C'\mathbf{I})$ to avoid instabilities. The pseudo-inverse, $\mathbf{A}^+$, of a matrix **A** can be obtained by the singular value decomposition of **A**. Any real matrix $\mathbf{A}_{m\times n}$ can be expresses as:

$$\mathbf{A}_{(m\times n)} = \mathbf{U}_{(m\times m)}\ \Sigma_{(m\times n)}\mathbf{V}_{(n\times n)}^T$$
$$= [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m] \times \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_m) \times [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]^T \qquad (3.48)$$

where **U** and **V** are orthogonal matrices and Σ is a diagonal matrix.

The pseudo-inverse $\mathbf{A}^+$ can then be obtained as:

$$\mathbf{A}^+_{(n\times m)} = \mathbf{V}_{(n\times n)}\,\Sigma^+_{(n\times m)}\,\mathbf{U}_{(m\times m)}^T \qquad (3.49)$$

where $\Sigma^+ = \text{diag}(1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_r, \mathbf{0})$.

There are three parameters of this scheme: (1) σ, the width of the Gaussian kernel; (2) q, the number of good representative points or the number of dimension of the approximated features; and (3) C′, ridge regression coefficient. These three parameters will be calibrated together with the embedding structure parameters (τ, d) with the minimum prediction error used as the objective function.

The linear regression applied to the target variable and the approximated features is theoretical based and its algorithm is reliable. The approximation is based on the eigenvalue and eigenfuntion of the eigenvalue problem of the Mercer’s theorem. The selection of points for the features approximation is based on the entropy which suggests most representative points. Linear regression is a very simple and fast operation. Therefore the computation time of this scheme can be sure to be short for chaotic time series analysis with large data record.

3.5 Summary and conclusion

In this Chapter, SVM has been proposed to be applied in the phase space and in the dynamics reconstructions. Applying SVM as a regression engine, the parameters of embedding structure and SVM are calibrated with minimum prediction error as the objective function.

Computational difficulty with large data records faced by SVM was discussed. Effective and efficient techniques to overcome this are called for. Two techniques are offered. These techniques are separately coupled with SVM to make SVM of practical value in analysing chaotic time series. One of the techniques is the decomposition method, which decomposes the quadratic programming problem of a large number of variables into a series of small quadratic programming problems each with 2 variables at a time. The other technique is a linear ridge regression carried out directly between the target variable and the features.

Parameter calibration is required. There are altogether 5 calibration parameters. Two are from the embedding structure while three are from SVM with the selected kernel. Like any other models the proposed SVM will perform best when the parameters are well calibrated. In this study, the calibration is done automatically with an evolutionary algorithm.

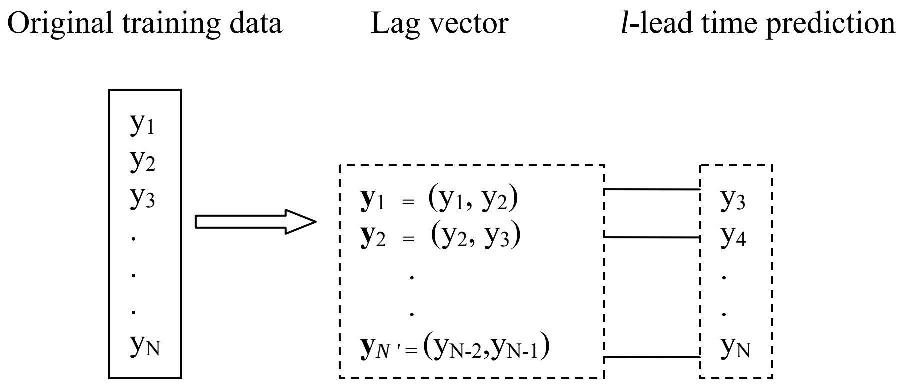


Figure 3.1 Reconstructed phase space data set with (τ =1, d=2, l=1)

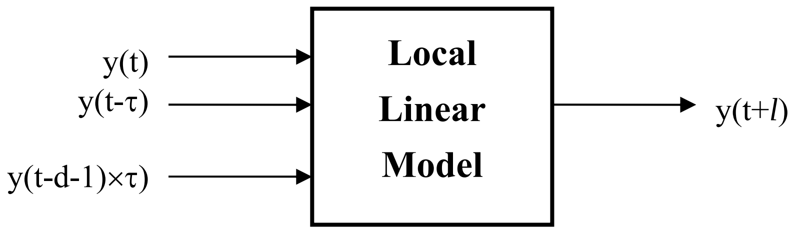


Figure 3.2 Architecture of local model for dynamics reconstruction

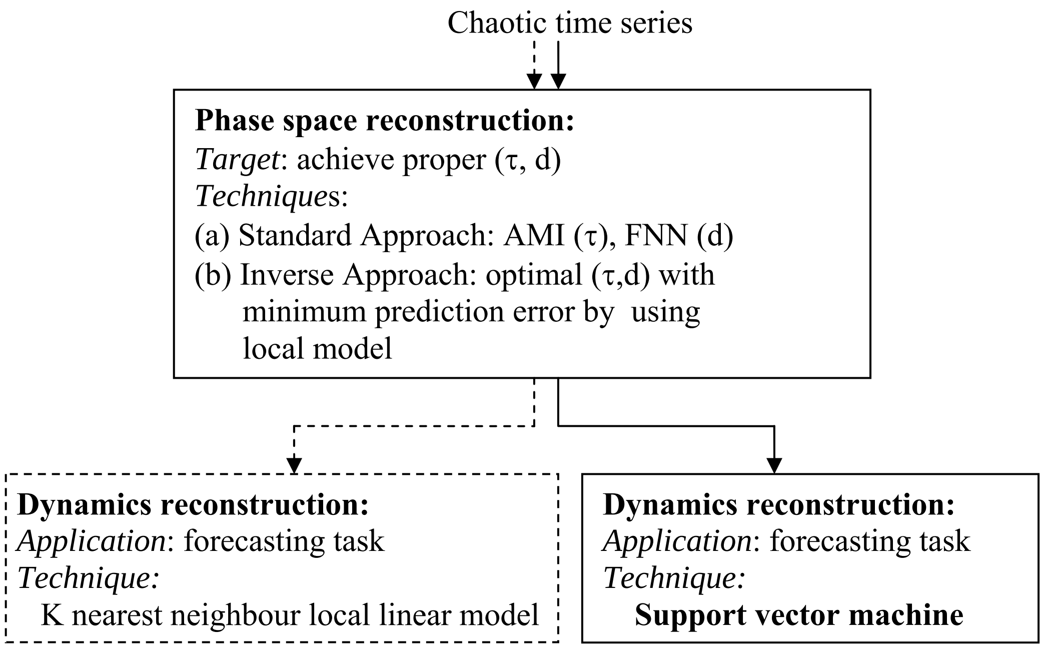

Chaotic time series
Phase space reconstruction:
Target: achieve proper (τ, d)
Techniques:
(a) Standard Approach: AMI (τ), FNN (d)
(b) Inverse Approach: optimal (τ,d) with minimum prediction error by using local model
Dynamics reconstruction:
Application: forecasting task
Technique:
K nearest neighbour local linear model
Dynamics reconstruction:
Application: forecasting task
Technique:
Support vector machine

Figure 3.3 Architecture of SVM for dynamics reconstruction

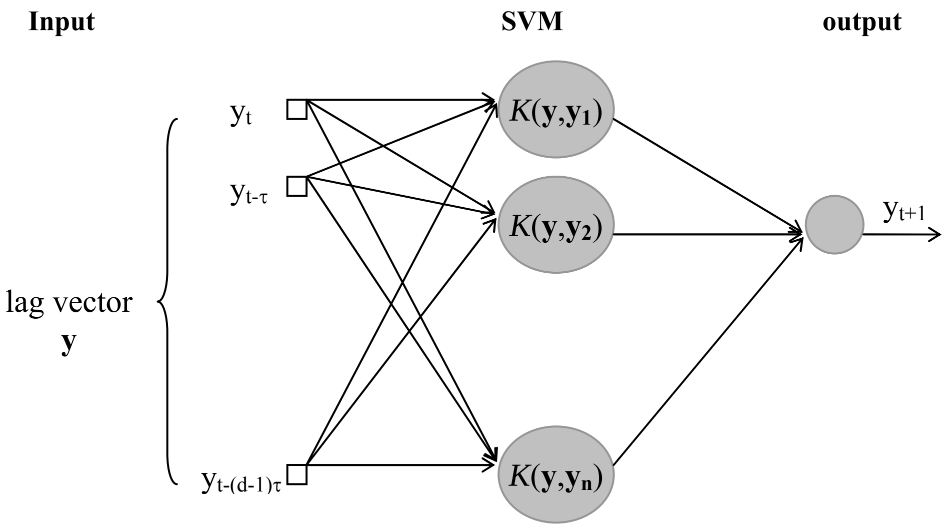

Input
SVM
output
y_t
$y_{t-\tau}$
$y_{t-(d-1)\tau}$
lag vector **y**
$K(\mathbf{y},\mathbf{y_1})$
$K(\mathbf{y},\mathbf{y_2})$
$K(\mathbf{y},\mathbf{y_n})$
y_{t+1}

Figure 3.4 Diagram of dynamics reconstruction of chaotic time series

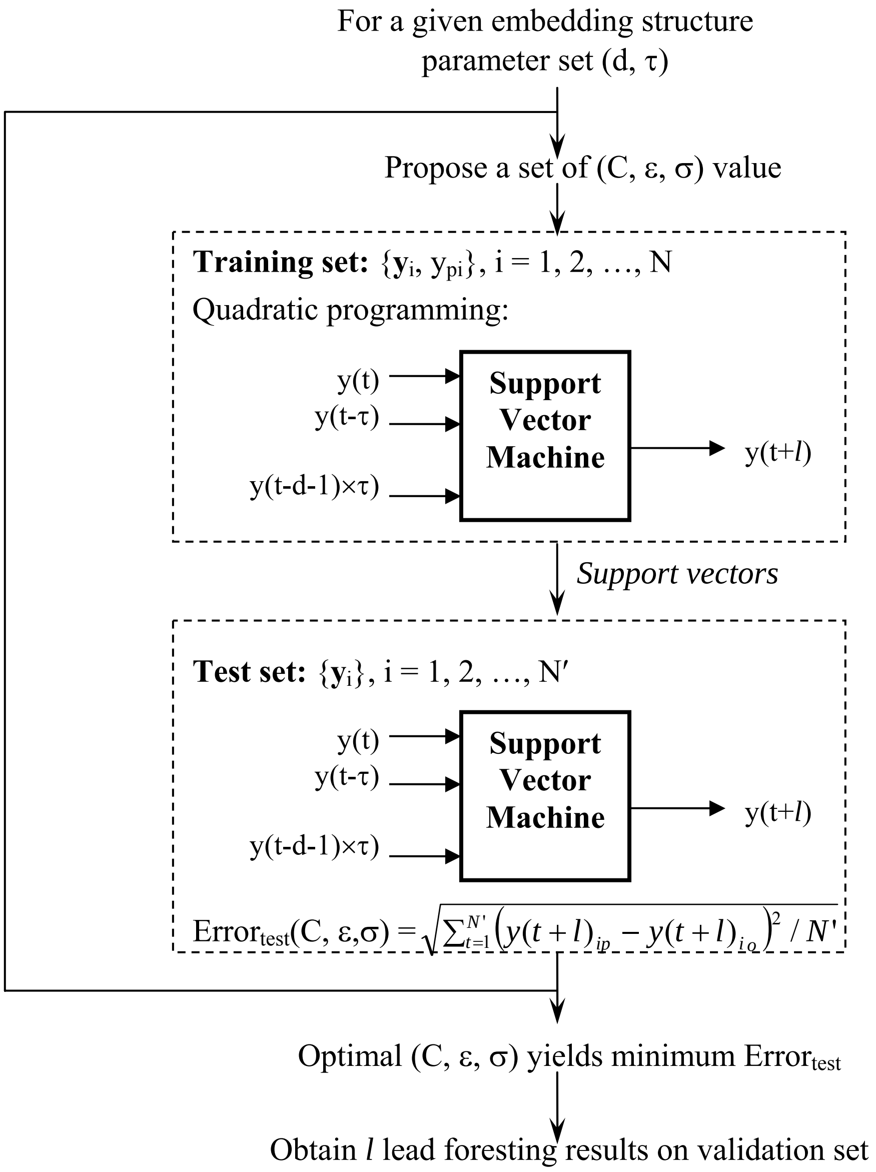
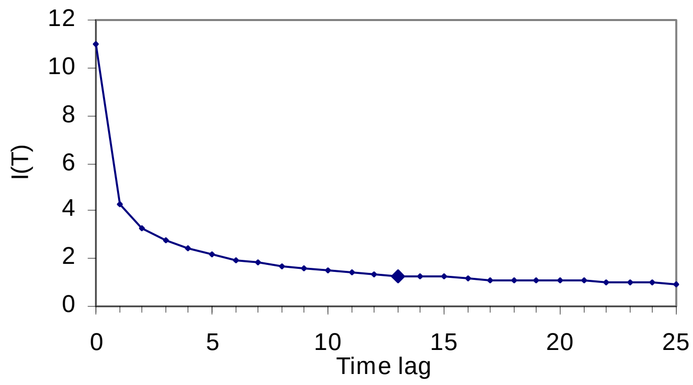
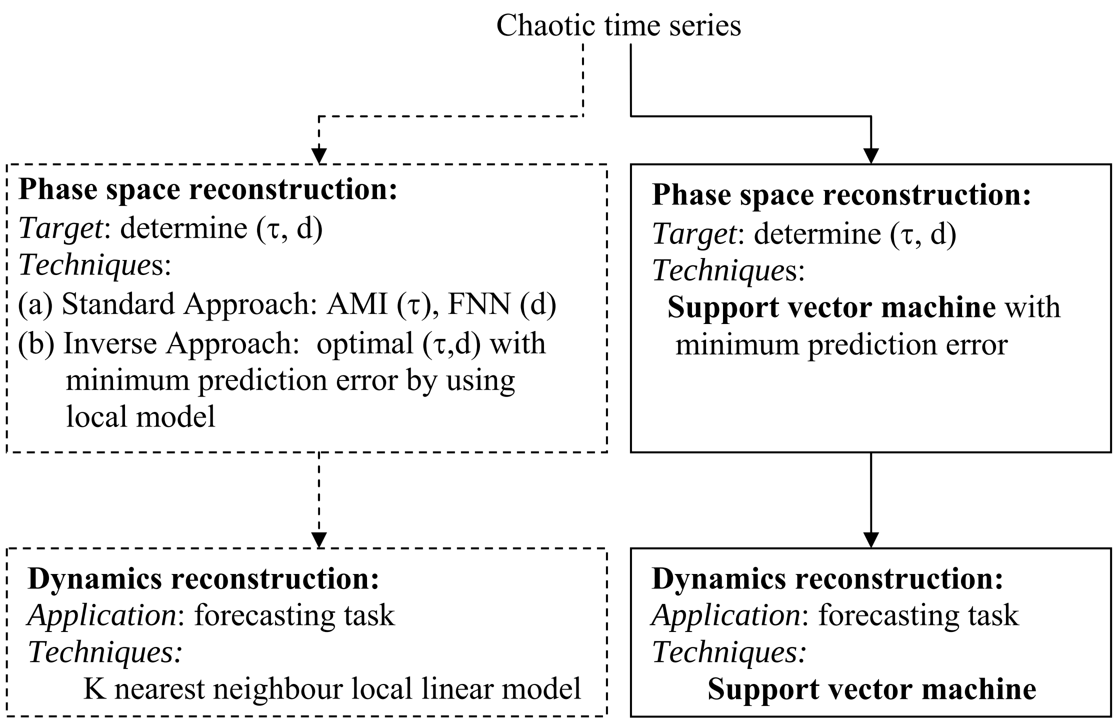

For a given embedding structure parameter set (d, τ)
Propose a set of (C, ε, σ) value
Training set: {yi, ypi}, i = 1, 2, …, N
Quadratic programming:
y(t)
y(t-τ)
y(t-d-1)×τ)
Support Vector Machine
y(t+l)
Support vectors
Test set: {yi}, i = 1, 2, …, N′
y(t)
y(t-τ)
y(t-d-1)×τ)
Support Vector Machine
y(t+l)
Error_test(C, ε,σ) = √(Σ_{t=1}^{N'} (y(t+l)_ip − y(t+l)_io)² / N')
Optimal (C, ε, σ) yields minimum Error_test
Obtain l lead foresting results on validation set

Figure 3.5 Schematic diagram of proposed SVM parameter set selection



Time lag	I(T)
8	1.68008
9	1.60418
10	1.50057
11	1.44219
12	1.33023
13	**1.27403**
14	1.28134
15	1.2665
16	1.17871
17	1.07548

Figure 3.6 Average mutual information (AMI) and time lag selection



Chaotic time series
Phase space reconstruction:
Target: determine (τ, d)
Techniques:
(a) Standard Approach: AMI (τ), FNN (d)
(b) Inverse Approach: optimal (τ,d) with minimum prediction error by using local model
Phase space reconstruction:
Target: determine (τ, d)
Techniques:
Support vector machine with minimum prediction error
Dynamics reconstruction:
Application: forecasting task
Techniques:
K nearest neighbour local linear model
Dynamics reconstruction:
Application: forecasting task
Techniques:
Support vector machine

Figure 3.7 Parameters determination and task performances with differences techniques: Standard, Inverse, and SVM approaches

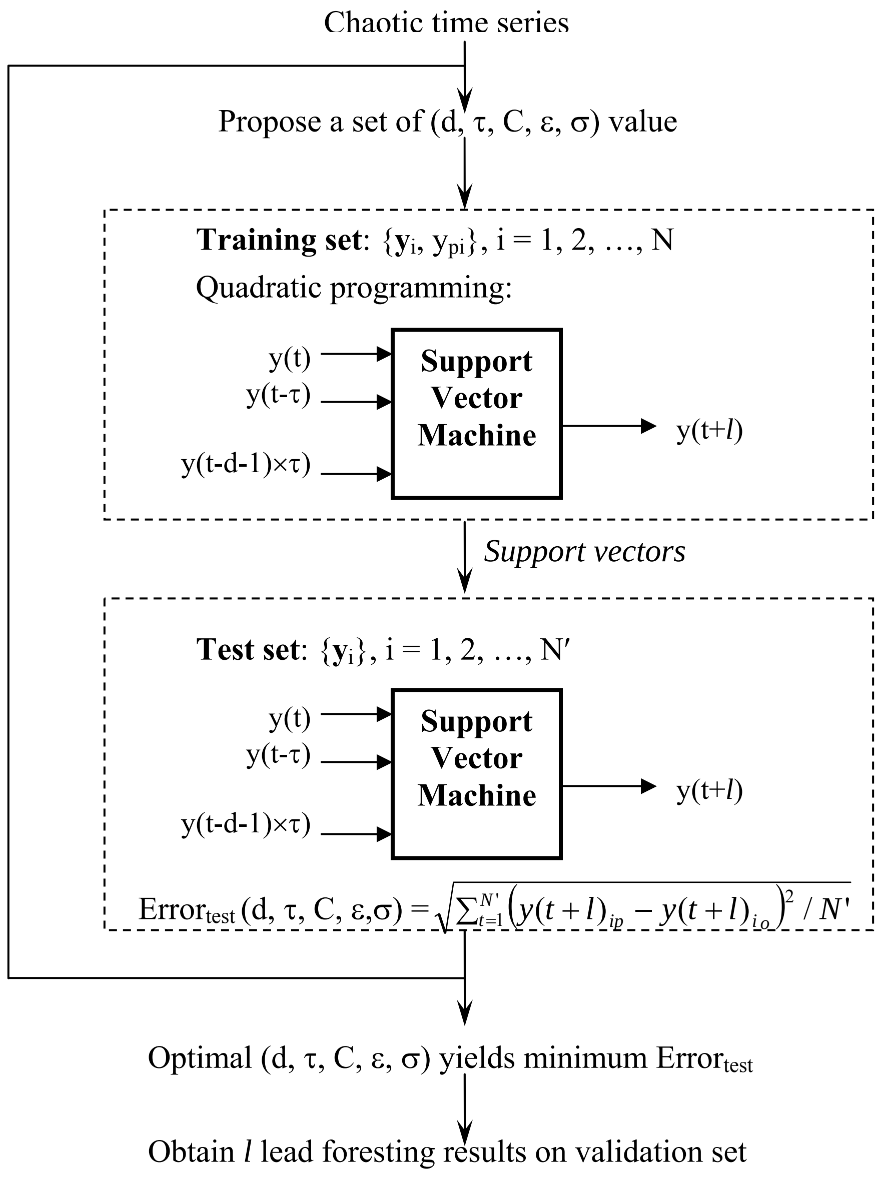

Chaotic time series
Propose a set of (d, τ, C, ε, σ) value
Training set: {yi, ypi}, i = 1, 2, …, N
Quadratic programming:
y(t)
y(t-τ)
y(t-d-1)×τ)
Support Vector Machine
y(t+l)
Support vectors
Test set: {yi}, i = 1, 2, …, N′
y(t)
y(t-τ)
y(t-d-1)×τ)
Support Vector Machine
y(t+l)
Errortest (d, τ, C, ε,σ) = √(Σ (y(t+l)ip − y(t+l)io)² / N')
Optimal (d, τ, C, ε, σ) yields minimum Errortest
Obtain l lead foresting results on validation set

Figure 3.8 Schematic diagram of SVM for phase space and dynamics reconstruction

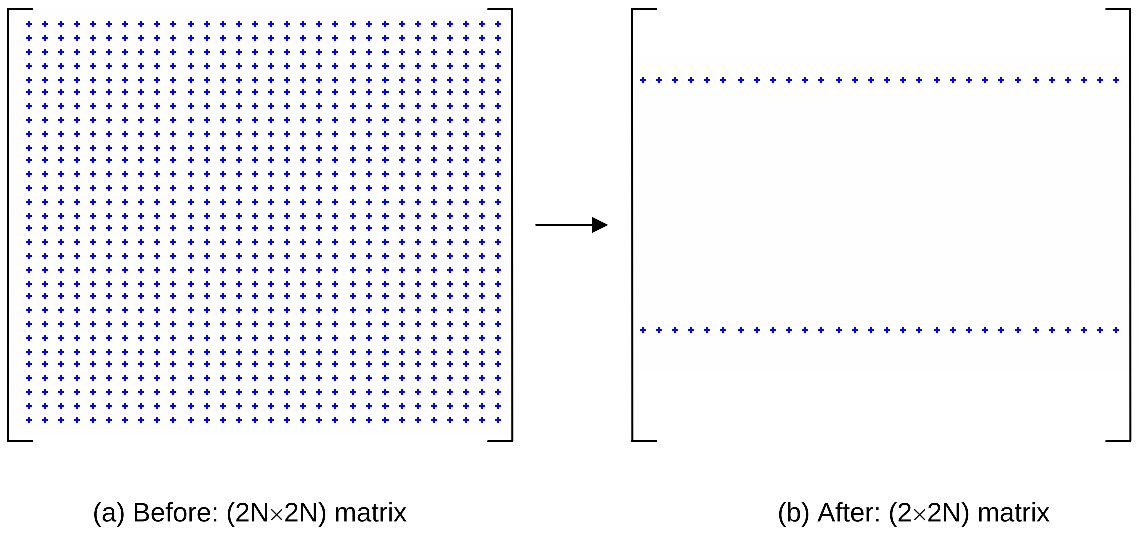


Figure 3.9 Illustration of memory requirement for quadratic programming before and after decomposition scheme

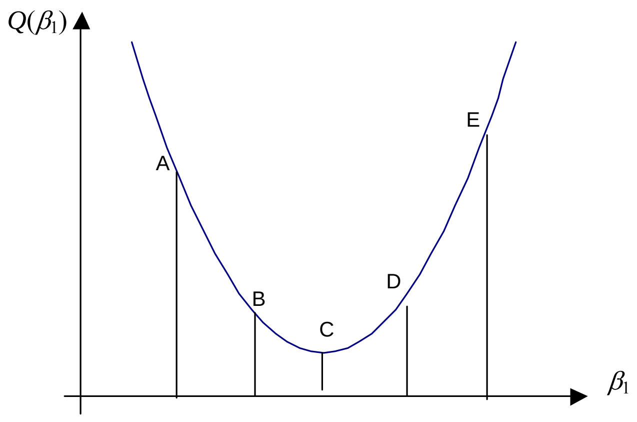
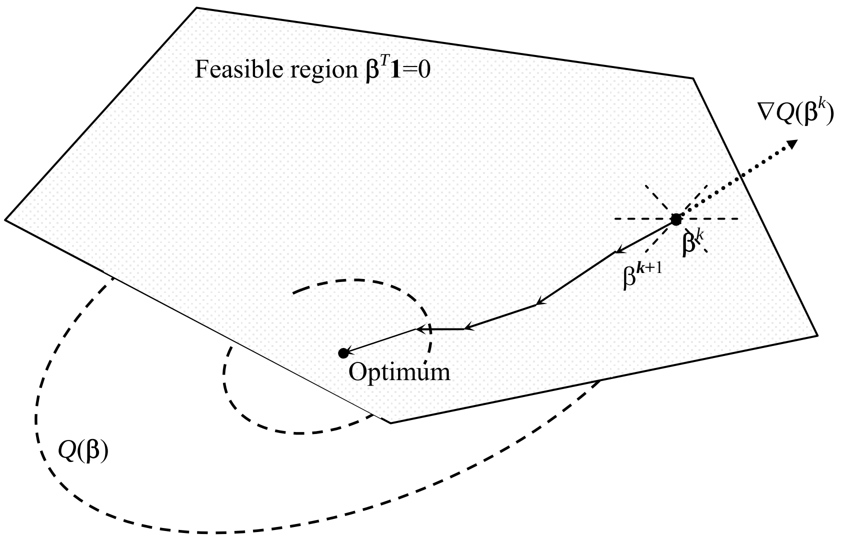
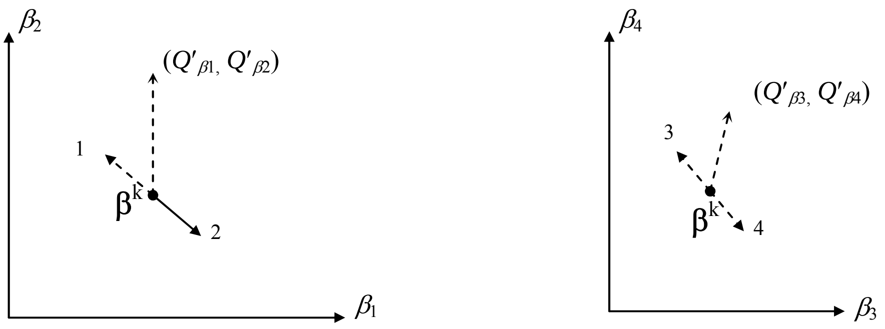


Figure 3.10 SVM decomposition optimization problem with working set of 2 variables



Feasible region $\boldsymbol{\beta}^T\mathbf{1}=0$
$\nabla Q(\boldsymbol{\beta}^k)$
$\boldsymbol{\beta}^k$
$\boldsymbol{\beta}^{k+1}$
Optimum
$Q(\boldsymbol{\beta})$

(a) Progression of decomposition algorithm



β_2
$(Q'_{\beta 1}, Q'_{\beta 2})$
1
β^k
2
β_1
β_4
$(Q'_{\beta 3}, Q'_{\beta 4})$
3
β^k
4
β_3

(b) Selection of working variables

Figure 3.11 Illustration of decomposition method in SVM quadratic programming

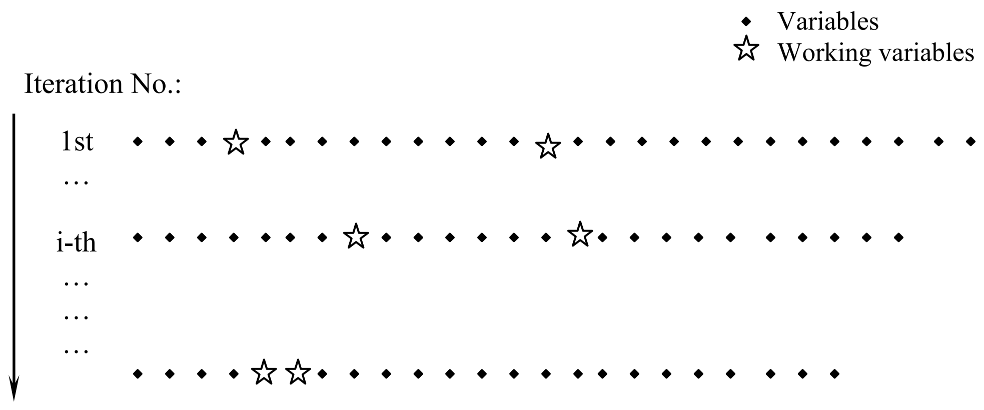
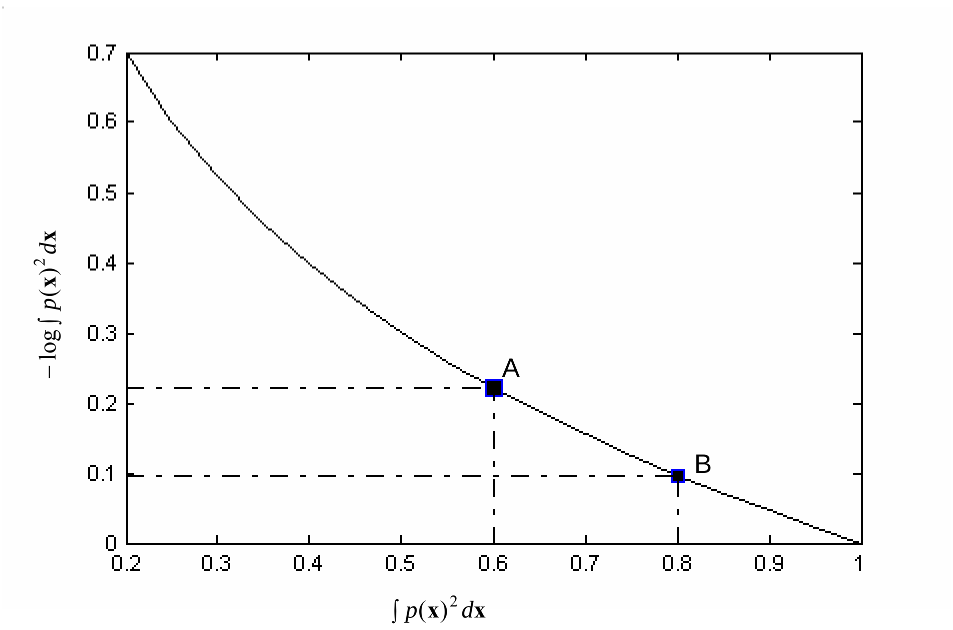
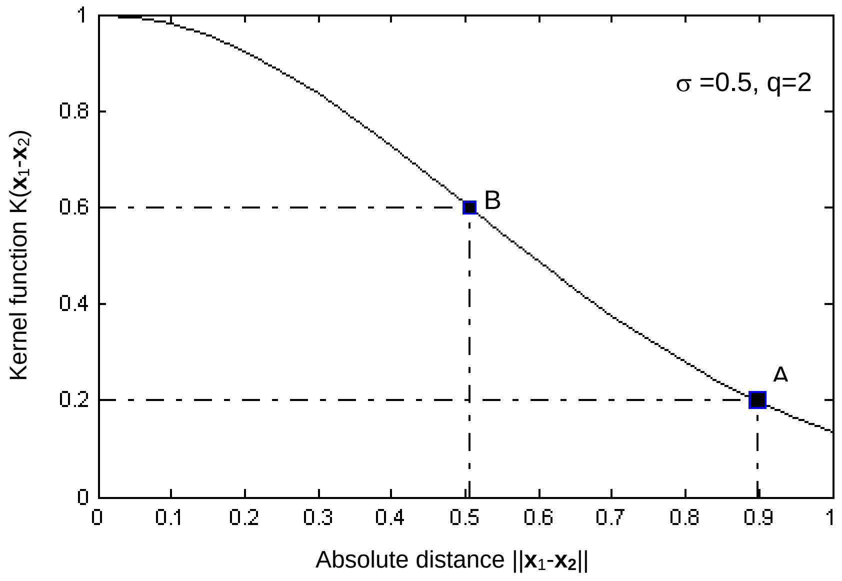


Figure 3.12 Illustration of shrinking process (reducing number of variables) in decomposition algorithm



(a) Renyi entropy function



(b) Kernel function

Figure 3.13 Illustration of quadratic Renyi entropy function and scatter

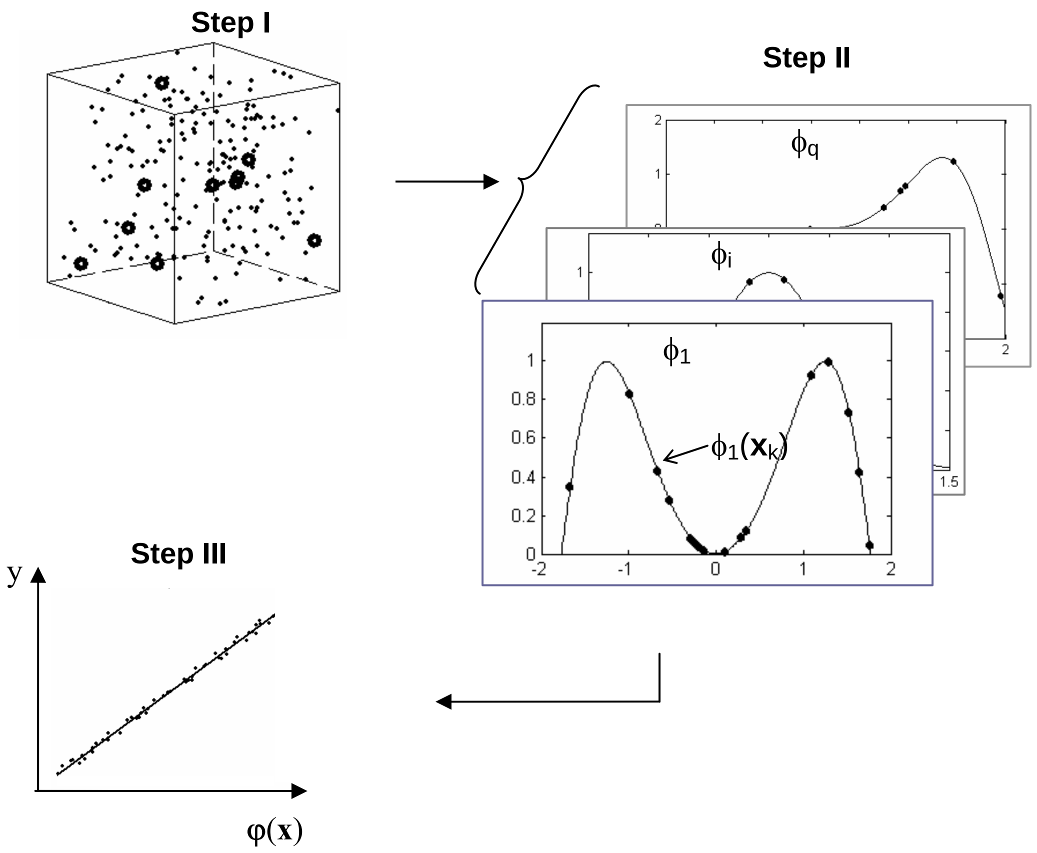


Figure 3.14 Schematic diagram of ridge regression in feature space

CHAPTER 4
PARAMETER CALIBRATION WITH EVOLUTIONARY ALGORITHM

4.1 Introduction

The need for an effective and efficient optimization scheme to calibrate the SVM parameters and the embedding structure parameters was discussed in chapter 3. There are a total of five parameters to be determined in this approach. They are: the time delay (τ), the embedding dimension (d), and the three SVM parameters (C, ε, σ) for the decomposition method and (C′, q, σ) for the ridge regression method.

The range of these five parameters may be as follows: τ from 1 to 20 (with increment 1), d from 2 to 21 (with increment 1), C′ from 0.1 to10 (with increment 0.1), q from 20 to 100 (with increment 1), and σ from 0.1 to 0.9 (with increment 0.01). There are a total of 256×10^6 (= $20\times20\times100\times80\times80$) possible combinations. A brute-force search method is certainly not efficient. Instead, an efficient evolutionary algorithm is of interest for exploration.

Evolutionary Algorithms have been reported to effectively and efficiently yield the optimal solution within the search range. The proposed approach, EC-SVM, which couples SVM with an Evolutionary algorithm applied to Chaos based reconstructed phase space is described in this chapter. In addition, EC-SVM, as described in Chapter 3, is a SVM equipped with the decomposition method or the linear ridge regression to deal with large data size.

4.2 Evolutionary algorithms for optimization

4.2.1 Introduction

Evolutionary Algorithms (EAs) are the common term used for algorithms based on principles of nature (evolution, genetic). Evolutionary Algorithms cover genetic algorithms, evolution strategies, evolutionary programming and genetic programming. Different evolutionary algorithms evolved during the last 35 years: genetic algorithms developed by Holland (1975), evolutionary strategies developed by Rechenberg (1973) and Schwefel (1981), and evolutionary programming by Fogel, et al. (1966).

Unlike classical optimization techniques, evolutionary algorithm is a population-based stochastic search and optimization technique. Most classical optimization methods generate a deterministic sequence of iterative solutions based on the gradient or high order statistics of the cost function. In EAs, it is not necessary to require gradient or other auxiliary information; only an objective function or multi objective functions are required.

Moreover, most of the classical technique often ends up at local optimal solution. EAs work with a population of points instead of a single point. EAs have been shown to outperform classical methods and can tackle difficult optimization tasks of the real world problems where classical techniques are not applicable or fail to provide satisfactory solutions.

A collection of solutions called current population is updated by replacing part of the population by offspring. There are various types of genetic representation of solutions to the problem, binary encoding or real number encoding. The population evolves into next generation by a series of processes such as selection, reproduction, and mutation. The fitness function value is a criterion to judge if a solution is a good individual.

An initial population is first randomly generated. The algorithm evolves through fitness evaluation, selection, reproduction, mutation, etc. which are likely to create even better individuals for the next generation. The selection chance of each individual depends on its fitness. The fitter the individual is, the higher it is to be selected and, thus, its genes will be passed on to the next generation. If the optimization criteria are met, the final solution is the best solution among the population. The process is illustrated in Fig. 4.1.

Selection process is inspired by the role of natural selection in evolution — an evolutionary algorithm performs a selection process in which the fittest members of the population survive, and the least fit members are eliminated. In a constrained optimization problem, the notion of fitness depends partly on whether a solution is feasible (i.e. whether it satisfies all of the constraints), and partly on its objective function value.

The selection process is the step that guides the evolutionary algorithm towards ever-better solutions. Selection determines which individuals are chosen for mating (recombination) and how many offspring each selected individual produces. The first step is fitness assignment, e.g. by proportional fitness assignment. The actual selection is performed in the next step. Parents are selected according to their fitness values by means of one of the following algorithms, e.g. roulette-wheel selection, or tournament selection.

Reproduction is the process to generate offspring from the last generation and is accomplished through transfer of the genes. There are various types of reproduction such as crossover for binary code or recombination for real code. The new offspring created from this process form a part of the population in the next generation.

In mutation process the genes of one or more members of the current population are mutated to yield a new population. The new solution may be better or worse than the population member whose genes are mutated. Its main purpose is to maintain diversity within the population and inhibit premature convergence. Mutation alone induces a random walk through the search space.

Since EAs are stochastic in nature in its search for optimal solution, it is difficult to specify exactly the convergence criteria. A common practice is to stop GA after a fixed number of generations or if the performances of the best solutions insignificantly different.

For the parameters calibration problem in this study, the optimal set of (τ, d, C, ε, σ) or (τ, d, C', q, σ) is the set which yields the least prediction error when applied on the test data set. It should be noted that the search values of τ, d, q are integers while those of C, C', ε, σ are real numbers.

4.2.2 Shuffled Complex Evolution

The parameter search scheme used in this study is the Shuffled Complex Evolution (SCE) algorithm. The SCE method was developed at the University of Arizona (Duan et al., 1992). It is a hybridisation of several salient features of several optimisation techniques and has been demonstrated in various studies to be a robust and efficient technique.

4.2.2.1 Description of algorithm

The SCE algorithm is based on the synthesis of four concepts:

(1) Combination of probabilistic and deterministic approaches: using probability to determine survivability;

(2) Clustering: the shuffling of complexes and sharing of information in each complex;

(3) Systematic evolution: to ensure global improvement; and

(4) Competitive evolution: to ensure the competitiveness of the fittest.

The SCE optimisation method combines the best features of complex shuffling and evolution and attempts to locate the global optimum, using the strength of the local optimisation simplex procedure (Nelder and Mead, 1965) with the idea of a controlled random search and complex shuffling (Duan et al., 1992).

The method begins with a population of points sampled from the feasible space. The population can be partitioned into one or more communities. Each community evolves based on a statistical 'reproduction' process that uses the 'simplex' geometric shape to direct the search in the correct direction. At periodic stages in the evolution, the entire population is shuffled and points are reassigned to communities to ensure information sharing. As the search progresses, the entire population tends to converge toward the neighbourhood of the global optimum, provided the initial population size is large (Duan et al. 1992).

In essence, the SCE algorithm is a search algorithm for the global optimum. It directs its search in a principled manner as described in detail below.

(1) Generate sample – sample s points in the feasible parameter space and compute the objective function or criterion value for each point. The samples are generated randomly within the search range;

(2) Rank points – sort the s points in order of increasing criterion value so that the first point represents the smallest criterion value and the last point represents the largest criterion value (the goal is to minimise the criterion value);

(3) Partition into complexes – partition the s points into p complexes, each containing m points. The complexes are partitioned such that the first complex contains every p(k–1)+1 ranked point, the second complex contains every p(k–1)+2 ranked point, and so on, where k = 1, 2, …, m;

(4) Evolve each complex – evolve each complex according to the competitive complex evolution (CCE) algorithm, which will be elaborated later in this section;

(5) Shuffle complexes – combine the points in the evolved complexes into a single sample population at a defined stage of the evolution; sort the sample population in order of increasing criterion value; shuffle (i.e. re-partition) the sample population into p complexes according to the procedure specified in Step (3);

(6) Check stopping criteria – if any of the stopping criteria are satisfied, stop; otherwise continue. The search will cease when the stopping criteria is satisfied, it would continue otherwise;

(7) Check the reduction in the number of complexes – if the minimum number of complexes required in the population, p_{min}, is less than p, remove the complex with the lowest ranked points; set p = p–1 and s = p × m; return to step (4). If p_{min} = p, return to step (4).

The initial sampling of the parameter space provides the potential for locating the global optimum without being biased by the pre-specified starting points. The partition of the population into several communities facilitates a freer and more extensive exploration of the feasible space in different directions, thereby allowing the possibility that the problem has more than one region of attraction. The shuffling of communities enhances the survivability by sharing of the information (about the search space) gained independently by each community. The SCE search algorithm is summarised in Fig. 4.2.

4.2.2.2 Competitive Complex Evolution

The key component of the SCE method is the competitive complex evolution (CCE) algorithm. This algorithm is based on the Simplex Downhill Search scheme of Nelder and Mead (1965). Each evolution on the complex generates a new offspring by using the operations of selection, reflection, contraction and mutation.

Selection is based on the fitness and individuals with high fitness values have higher probabilities to be chosen. The offspring generated replaces the worst points in the complex. If the offspring generated by reflection is failed to be better than the worst individual, then a contraction process is used to generate an offspring. If the offspring generated fails to perform better than the worst individual, an offspring is randomly generated.

Figure 4.3 illustrates the basic processes of reflection and contraction in two dimensions. For minimization problem, **G** is the worst point, i.e. $f_G > f_M > f_S$. The centriod of the points **X**, excluded the worst point **G**, can be calculated from:

$$\mathbf{X} = (\mathbf{M}+\mathbf{S})/2, \tag{4.1}$$

For general cases with q points, $\mathbf{U}_i$, i=1, 2, …, q, in the complex and in high dimension,

$$\mathbf{X} = \frac{1}{q-1}\sum_{i=1}^{q-1}\mathbf{U}_i \tag{4.2}$$

For the reflection point **R**, **X** is the centre point of **R** and **G**, i.e. **X** = (**R** + **G**) / 2. **R** can be calculated as:

$$\mathbf{R} = 2\mathbf{X} - \mathbf{G} \tag{4.3}$$

The contraction point **C** is the centre point of **X** and **G**, i.e.

$$\mathbf{C} = (\mathbf{X} + \mathbf{G}) / 2 \tag{4.4}$$

The scheme is as follows:

(1) Construct a sub-complex by randomly selecting q points from the complex according to a triangular probability distribution. The probability distribution is specified such that the best point has the highest chance of being chosen to form the sub-complex, and the worst point has the least chance.

(2) Identify the worst point of the sub-complex and compute the centroid of the sub-complex without the inclusion of the worst point.

(3) Do a reflection step by reflecting the worst point through the centroid. If the newly generated point is within the feasible space, go to Step (4); otherwise, go to Step (5).

(4) If the newly generated point is better than the worst point, replace the worst point by the new point. Go to Step (7). Otherwise, go to Step (5).

(5) Do a contraction step by computing a point halfway between the centroid and the worst point. If the contraction point is better than the worst point, replace the worst point by the contraction point and go to Step (7). Otherwise, go to Step (6).

(6) Randomly generate a point within the feasible space. Replace the worst point by the randomly generated point.

(7) Repeat Steps (2) – (6) α times, where $\alpha \geq 1$ is the number of consecutive offspring generated by each sub-complex.

(8) Repeat Steps (1) – (7) β times, where $\beta \geq 1$ and β is the number of evolution steps taken by each complex.

4.2.2.3 Control parameters and stopping criteria

The SCE method contains many probabilistic and deterministic components that are controlled by some control parameters. The control parameters are:

(1) p, the number of complexes;

(2) m, the number of points in a complex;

(3) q, the number of points in a sub-complex;

(4) p_{min}, the minimum number of complexes required in the population;

(5) α, the number of consecutive offspring generated by each sub-complex; and

(6) β, the number of evolution steps taken by each complex.

The number of the initial randomly generated sampling points, s, is a product of the number of complexes and number of points in a complex (= m $\times$ p). It was recommended by Duan et al. (1992) that the chosen m value should be such that m = 2n+1. n is the number of parameters to be optimised. The number of calibration parameters in this study is 5. The values for the control parameters stemmed from the recommended values by Duan et al. (1992), are summarised in Table 4.1.

The stopping criteria for the search algorithm are:

(1) The population has converged to pre-specified value of the original parameter space;

(2) The relative change in the objective function within the last *k* shuffling loops has not changed more than a pre-specified percentage; and

(3) The total number of evaluations has exceeded a pre-defined value.

4.3 EC-SVM I: SVM with decomposition algorithm

EC-SVM I is SVM equipped with the decomposition algorithm to solve large data sets in analysis of chaotic time series. SVM is applied in phase space reconstruction and in dynamics reconstruction. A quadratic programming problem with large variables is transformed into a series of quadratic programming problem each with 2 variables only. The parameters of the embedding structure (τ, d) and the SVM parameters (C, ε, σ) are calibrated automatically with shuffled complex evolution. The optimal set for these 5 parameters is the set which yields the least prediction error.

4.3.1 Introduction

Much have been described (in chapter 2) about various ways, AMI and FNN for examples, to derive 'reasonable' values for the parameters of the embedding structure (τ, d). The associated problems with the derived (τ, d) values have also been discussed in chapter 2. In this section focus is placed on methods/recommendations, suggested in literatures, to select the SVM parameters (C, ε, σ). Selection of SVM parameters remains a difficult task and some recommended methods are summarized in the following:

(1) Since SVM model complexity strongly depends on the number of support vectors, Schölkopf et al. (1998) suggest to use another control parameter ν (which represents a fraction of support vectors instead of ε. In this approach, parameter ν has to be user-defined. Similarly, Mattera and Haykin (1999) proposed to choose ε-value so that the number of support vectors is around 50% of the number of samples. Many problems show that optimal generalization performance is achieved when the number of support vectors is significantly different from 50%.

(2) Smola et al. (1998a) and Kwok (2001) proposed that the optimal ε-value is asymptotically proportional to the noise variance. The higher the noise variance is, the higher value ε should be. The pitfall result is that their practical value is limited only to cases when the noise level is known or can be estimated. However, noise variance is rather difficult to be satisfactorily estimated in real data. There us, however, no practical guideline as to how to estimate noise level satisfactorily for real world time series.

(3) Mattera and Haykin (1999) proposed to choose to select parameter C about equal to the range of output values. The suggestion is only good to avoid numerical instability. However, case studies show that when C is significantly larger or smaller than the range of output data it outperforms the situation when C is set as the range of the output value.

(4) Use of cross-validation for parameter selection. As illustrated in the Section 4.1, this is very computation and data-intensive and the scheme is feasible only for a limited number of values.

(5) Cherkassky and Ma (2004) provided statistical motivated approach to the selection of C as 3 times the standard variance and ε depends on the record length and variance. σ is as 0.2-0.5 of the range of the input data. However, the empirical formula of ε include another empirical coefficient which needs be assigned. The suggested value is not a set of unique value. Users still need to adjust the parameters from a certain suggested region which, however, does not guarantee that the best selection resides in this region.

Even though there are a number of methods proposed by various researchers as summarized above, the parameter selection still remains a difficult problem unsolved. None of the above methods is perfect and guarantees a good performance for real world problems. As it can be seen, these methods are empirical in nature. Moreover, the choices could not be a single set of parameter and tuning task of selecting the parameters still remains.

4.3.2 Calibration parameters

SCE is the search engine used in this study to find the optimal parameters representing the embedding structure and SVM equipped with decomposition method. There are 5 calibration parameters. They are:

(1) τ: time lag;

(2) d: embedding dimension;

(3) C: trade-off between complexity of the machine and the empirical error;

(4) ε: Insensitive zone in the SVM transformation; and

(5) σ: kernel parameter in the Gaussian kernel.

The objective function used is the mean squared error (MSE) of the test data set. A flow chart of the suggested search scheme is presented in Fig. 4.4.

The search is stopped if any of the following criteria is met. Meeting one of the criteria implies the convergence of the evolutionary algorithm.

(1) Population is converged into a small zone, e.g. 0.001 of the search space;

(2) Change of objective function value is negligible, e.g. less than 0.001 in the last 5 generations, for example.

In this study the total number of evaluations is set at a very large value, 2000. This large number of evaluations prevents the search from stopping before one of the above listed criteria is achieved.

4.3.3 Parameter range

Setting the range of each parameter is quite crucial in the search for optimal set. A wide range will take the search engine longer time to arrive at the optimal set. A small range, on the other hand, may risk missing the actual optimal value. Hence, a delicate balance in the choice of parameter range is required.

The 'optimal' value should not reside at the boundary of the parameter range. Should it, however, be the case, it implies that the real optimal may lie outside of the earlier defined range. Thus a new study with wider parameter range has to be conducted.

4.3.3.1 Parameters range of embedding structure

The average mutual information, AMI, is used as a guide for setting the upper limit of the time lag (τ) range. In this study the range for time lag is set at [1, 20]. It should be noted that the upper limit is higher than the value resulting from AMI.

The value of the embedding dimension (d) is traditionally suggested by the following studies:

(1) Takens Theorem (1981): $d = 2d_2 + 1$ where d_2 is the correlation dimension;

(2) Aberbanel et al. (1990); $d = d_2 + 1$; and

(3) Kennel et al. (1992) proposed the false nearest neighbour (FNN).

Similarly, the range of embedding dimension may start from 2 to a value reasonably higher than the value resulting from the above techniques. The range of the embedding dimension is set at [2, 20] in this study.

4.3.3.2 Parameter range of C in SVM

C yields a good trade off between the empirical error and model complexity. When Gaussian kernel (whose dimension is infinitely large) is used, C is particularly useful in balancing the complexity of the model and, at the same time, preventing the over fitting problem.

It is known that if C is chosen to be a very big value than the range of the output data, B, the numerical instability will occur as shown in Eq. (2.33) and Section 2.3.5. As to the decomposition method, a high C value may cause the method oscillation in the training and hence long computational time particularly when the kernel trick is employed for the SVM formulization.

Figure 4.5 demonstrates the effect of varying C values (with different other parameters) on the training time and on the test error. The forecasting variable is normalized into the range at [0 1]. The training time increases with increasing C value. The minimum test error, 310.85m^3/s, occurs when C=13.69 which is a value not so close to the upper bound (1) of the target data. Test error also varies with varying C values as shown in Fig. 4.5.

C is set slightly higher than the B value to avoid possible numerical instability. As it is shown in Fig. 4.6, the upper bound of training time is about 300 seconds, i.e. 5 minutes, which is much faster to converge than the case shown in Fig. 4.5. The minimum test error is 323.21m^3/s, as shown in Fig. 4.6(b). This test error is higher than that in Fig. 4.5. Thus, lower C value can provide short training time but the test error may be higher since fundamentally C does not have any restrictions.

4.3.3.3 Parameter range of ε in SVM

ε parameter is associated with the ε-insensitive loss function. ε value is proportional to the noise variance. The noise variance of real time series is difficult to be correctly estimated. ε can start from zero, where the noise level is quite low or clean data, to a high value where the data is noisy. In this study, the upper bound of ε is set at 10% of B, the upper bound of the target variable. The range of ε is at [0, 0.1B].

4.3.3.4 Parameter range of σ of Gaussian Kernel

Gaussian kernel function is defined as:

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}'\right\|^2 \right) \tag{4.5}$$

Figure 4.7 displays a graph of Gaussian Kernel for different σ values when $||\mathbf{x}-\mathbf{x}'||$ is at a range [0, 1.2]. If the input data is ranged from [0, 1], $||\mathbf{x}-\mathbf{x}'||$ is range from [0,1]. Figure 4.7 shows the decreasing pattern for varying σ values. It shows that σ ranging from 0.1 to 0.8 is more favourable than σ outside of this range. K decreases drastically fast as σ is less than around 0.1; this may lead SVM to over fit the data including the noise portion. K value, however, decreases very gradually as σ is larger than about 0.8; this may cause SVM powerless to detect the nonlinear relationship in the data set. In this study, the range of the Kernel width σ is therefore set at 0.1 - 0.8 of the input data range. It should be noted that this range is slightly wider than that proposed by Cherkassky and Ma (2004).

4.3.4 Implementation

In EC-SVM I there are three major modules: (1) SCE evolutionary algorithm written in FORTRAN; (2) SVM decomposition method (SVM Torch II) for regression problem of large data sets, written in C language running on Linux system; (3) linking part of these two modules: SCE and SVM, organized with a Shell scripts file.

Figure 4.8 shows the diagram of the implementation scheme of EC-SVM I. The whole algorithm of EC-SVM I is running on Linux operation system. The implementation of EC-SVM I is as follows:

(1) SCE acts as an outside loop and is the main program since it gives instructions to execute various iterations automatically, for example, about 1000 iterations of different chromosomes.

(2) For each chromosome, its fitness is calculated. Prior to fitness evaluation there are several measures to be taken. They are:

    (a) Create lag vector and the corresponding *l*-lead day forecasting vector;

(b) Train SVM with the decomposition method;

(c) Apply the trained SVM to the test data set to compute prediction error;

(d) Convert the prediction error to the fitness function.

(3) From a given chromosome till calculating the fitness function, the task is organized in a shell scripts file. Shell provides an easier interface to execute commands. A Shell file is very similar to a DOS .BAT file, except that the shell scripts have more available functions. The Shell file reads the chromosome that is generated in SCE, and use this set of parameters (τ, d, ε, C, σ) to solve the SVM by decomposition method. The trained SVM is then applied on the test data set to obtain the prediction error which is converted to the fitness value in SCE algorithm.

(4) Another C-language file is formed to create a data file containing lag vectors and the corresponding forecast vector. This data file is used for SVM regression.

(5) SCE follows the evolutionary algorithm by creating chromosomes. The chromosomes evolve based on the fitness function values. After new chromosomes are generated, SCE sends the command to the shell scripts file which in turn sends the command to the respective procedures in various C files and data files.

(6) SCE evolves till the stopping criteria are met. The optimal solution is the one which yields the least prediction error. The whole algorithm of EC-SVM I is run on Linux operation system.

Evolutionary algorithm helps to fulfil the search automatically, effectively and efficiently within the specified range.

4.4 EC-SVM II: SVM with linear ridge regression

Since the decomposition method in EC-SVM I is an iterative algorithm, the training time may be to long. A ridge regression, applied directly in the kernel feature space is introduced in Section 3.4.2. Ridge regression requires no iterative scheme to solve the dual quadratic programming and the problem is solved in its original prime formulation.

Similar to EC-SMV I, the objective function is the mean squared error (MSE) resulting from the test data set. A flow chart of the scheme is illustrated in Fig. 4.4. The main framework of EC-SVM II is basically the same as that described in Section 4.3. The differences are illustrated in the following subsections.

4.4.1 Calibration parameters

There are five parameters required to be calibrated simultaneously for EC-SVM II:

(1) τ: Time lag;

(2) d: Embedding dimension;

(3) σ: kernel parameter for the Gaussian kernel;

(4) q: number of dimension of approximated feature space; and

(5) C': ridge coefficient in ridge regression.

The first three parameters (τ, d and σ) are the same as that in EC-SVM I; the same search ranges are also used in EC-SMV II. The remaining two parameters, q and C', will be discussed in the following sections.

4.4.1.1 Parameter C'

C' is a ridge regression coefficient. The formalization of SVM regression problem shown in Eq. (2.24) stems from the ridge regression problem shown in Eq.

(3.41). It is a means to control the balance between bias and variance, which are two measures of the effectiveness of the approximated prediction function. Geman et al. (1992) highlighted the bias and variance issues.

Bias represents the inability of the approximated prediction function to estimate the exact function. If on the average, the model result differs from to the regression function, the model is then said to be biased. An unbiased estimator may, however, still have a large error if the variance is large.

When an estimator has a small bias and is substantially more precise than the unbiased estimator, it is a more preferred estimator since it has a larger probability of being close to the true parameter value. As shown in Fig. 4.9, estimator w is unbiased but imprecise; estimator w^b, however, has a small bias but is much more precise. The probability that w^b falls near the true value is much greater than that for w. Thus, w^b is much preferred than w.

A good approximated forecasting function should be accurate and not sensitive. Deliberately introducing bias is equivalent to restricting the range of the function for which a model can account. The resulting loss of flexibility makes the model less sensitive. In ridge regression, its aim is to minimize the cost function as in Eq. (4.5); this penalises large weights, i.e. to restrict the flexibility. In general, finding the ‘flattest’ linear function translates to the following:

$$\text{Minimize } L(\mathbf{w}) = \sum_{j=1}^{N}(y_{o\,j} - \sum_{i=1}^{q} w_i h_i\,(\mathbf{x}_j))^2 + C'\sum_{i=1}^{q} w_i^{\,2} \qquad (4.6)$$

The regularisation parameter C′ controls the balance between fitting the data and avoiding the penalty. Small C′ means that the data can fit tightly without causing a large penalty. The introduced bias favours solution involving small weights; the effect is to smooth the output since large weights are usually required to produce a highly variable output.

Ridge regression is carried out in the approximated feature space with finite number of dimensions. Any C′ value does not cause the numerical instability problem, unlike the scheme when kernel trick is used. C′ parameter in ridge regression is much less sensitive than in the decomposition method; it does not have the instability restrictions.

Figure 4.10 demonstrates the effect of varying C′ values on the training time and test error. Training time is not as significantly influenced by C′ value as that in EC-SVM I as shown in Figs. 4.5 and 4.6.

4.4.1.2 Parameter q

q is the number of points selected from the training data set to estimate eigenvalues and eigenfunctions. The number of the dimensions of the approximated features may be slightly lower than q value; reason being some eigenvalues, very close to zeros, are eliminated to avoid numerical instability.

The higher the value q is set, the larger are the computational time and the memory space required. High value of q causes large kernel matrix and high feature dimensions. It should be noted that high value of q does not necessarily translate to better prediction accuracy in the test data set although it results in lower error in training data set. Test error may increase when q is larger than a certain value; this happens in over fitting cases.

Training time is highly dependent on the number of the dimensions of the approximated features. Computational time is one of the factors used to judge the performance of an algorithm. Another important factor will be the prediction accuracy.

It is quite clear that if the q value is low, it is unlikely that the prediction accuracy can be satisfactory. As q value is further increased, however, the prediction accuracy reduces and the resulting computational time increases significantly. Therefore, it is important to select an appropriate q value which will yield a good model performance.

Figure 4.11 depicts the above scenarios. Training set error decreases as q increases till about q=70; the accuracy then decreases gradually when q is further increased. Test set error also decreases as q increases till about q=70; the error then increases particularly when q is greater than 100. The training time increases monotonously as q increases and it increases very rapidly particularly when q is larger than 100. From the above observations, the search range for q can be varied from 10 to slightly larger than 100, say 105 or 110.

Figure 4.12 shows an example that q has a range set at [10, 105]. The test error is sufficiently small and the training time is fast. This shows that the proposed range is quite reasonable.

4.4.2 Implementation

There are two modules in EC-SVM II: (1) SCE evolutionary algorithm which is written in FORTRAN; (2) Linear ridge regression in approximated feature space. The linear ridge regression is a very newly developed algorithm and only exists in MATLAB code, partly from LS-SVM lab (2002). The basic implementation strategy is the same as in Section 4.3.4. SCE acts as an outside loop and is the main program. It gives instructions to execute various iterations automatically about 1000 iterations of different chromosomes.

However, calling MATLAB application from FORTRAN for 1000 iterations is not very efficient and, at the same time, difficult for implementation; reasons are:

(1) In principle, MATLAB is good for developing the first stage of computational algorithm since it contains various packed functions. However, from the computational speed viewpoint, MATLAB is not as efficient as FORTRAN or C language, especially for applications containing several loops. It is common to use models written in C or FORTRAN; and MATLAB for the whole progress organization. To achieve higher computational performance, algorithms implemented in C or FORTRAN language are more efficient.

(2) Even though MATLAB has certain compatibilities with C or FORTRAN, these compatibilities are rather limited. Calling MATLAB from C or FORTRAN is complicated and not efficient. Since C and FORTRAN are much more fundamental computer languages, C and FORTRAN applications can be used freely and efficiently by other applications, or each other. Stand-alone applications, i.e. the executable files, can be run much easily and efficiently.

(3) MATLAB Compiler can convert some MATLAB applications to stand-alone C and C++ code. Special cares must be given to MATLAB files writing. Failures in such conversion are often encountered. Moreover, the C files converted from MATLAB are not very readable and it is difficult to make changes to suit to the applications.

(4) MATLAB engine library is a set of routines that make it possible to call MATLAB from other programs such as C or FORTRAN. There is a library of functions provided in MATLAB that allows starting and ending with MATLAB process, sending data to and from MATLAB, and sending commands to be processed in MATLAB. However, if the algorithm requires calling this piece of MATLAB application about 1000 times, relying on the engine library is not an efficient way. Time is spent mainly for this middle process instead of for computation.

Therefore, in this study the linear ridge regression with approximated features is firstly implemented in C and FORTRAN to suit the EC-SVM II algorithm. SCE, written in FORTRAN, calls this stand alone executable module. The scheme is shown in Fig. 4.13. The linear ridge regression should be stable and of high performance since it will be repeatedly called for about 1000 iterations. Its high quality must therefore be essential for the whole process.

In this study, there are mainly three steps, as shown in Fig. 4.14, for the implementation of the linear ridge regression with C or FORTRAN in the approximated feature space. The steps are:

(1) Select a good subset from the whole training set having the largest entropy. This is an easy implemented step. The algorithm is relatively simple;

(2) Approximate the feature space by using the eigenvalue and engenfunction of the kernel matrix from the selected subset. This step contains two procedures:

   (a) Eigendecomposion of the matrix $\mathbf{K}_{q\times q}$ to obtain its eigenvalue and eigenvectors; and

   (b) Feature estimation of the whole sample points. This can be fulfilled by using a loop going through all the training points.

(3) Apply ridge linear regression to fit the relationship between the target variable and the features. The essential part of this step is the pseudo-inverse operation, which can be calculated through singular value decomposition.

The eigendecomposition and singular value decomposition are the linear algebra operation. LAPACK, Linear Algebra PACKage, is written in Fortran77 and provides routines for solving eigenvalue problems, singular value problems, etc. LAPACK is also one of the FORTRAN libraries of MATLAB. MATLAB contains C code for linear algebra operations. Since they are packed and separated into several places in

MATLAB, it is not very convenient to abstract a small pack typically for a certain linear algebra operation as singular decomposition. The source codes for real matrices with double precision in FORTRAN are directly from LAPACK which is well implemented and has stable performances.

Thus, the linear regression in approximated feature space scheme implemented in C or FORTRAN in this study can be guaranteed a stable scheme, and, furthermore, EC-SVM II developed in this study can be guaranteed to be stable and yield high performance accuracy for large data sets in chaotic hydrological time series.

4.5 Summary

An evolutionary algorithm, SCE, is proposed to efficiently and systemically calibrate the parameters involved in a chaos based SVM technique. The SCE algorithm is first described followed by the implementation of this technique.

The novel approaches EC-SVM I and EC-SVM II are demonstrated in detail in this Chapter. Detailed implementation is elaborated for both decomposition method and linear ridge regression to overcome the large data sets problem. The search range selection of each parameter is demonstrated in detail.

Further, the performance of the proposed EC-SVM I and EC-SVM II will be demonstrated in two daily runoff time series. The performance will be conducted in the next chapter.

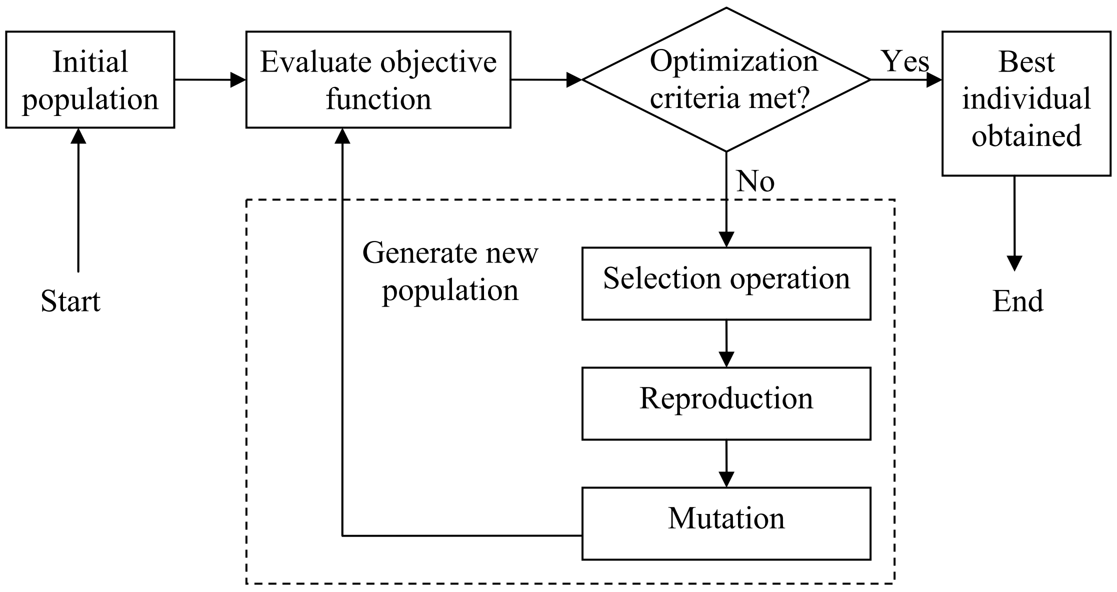
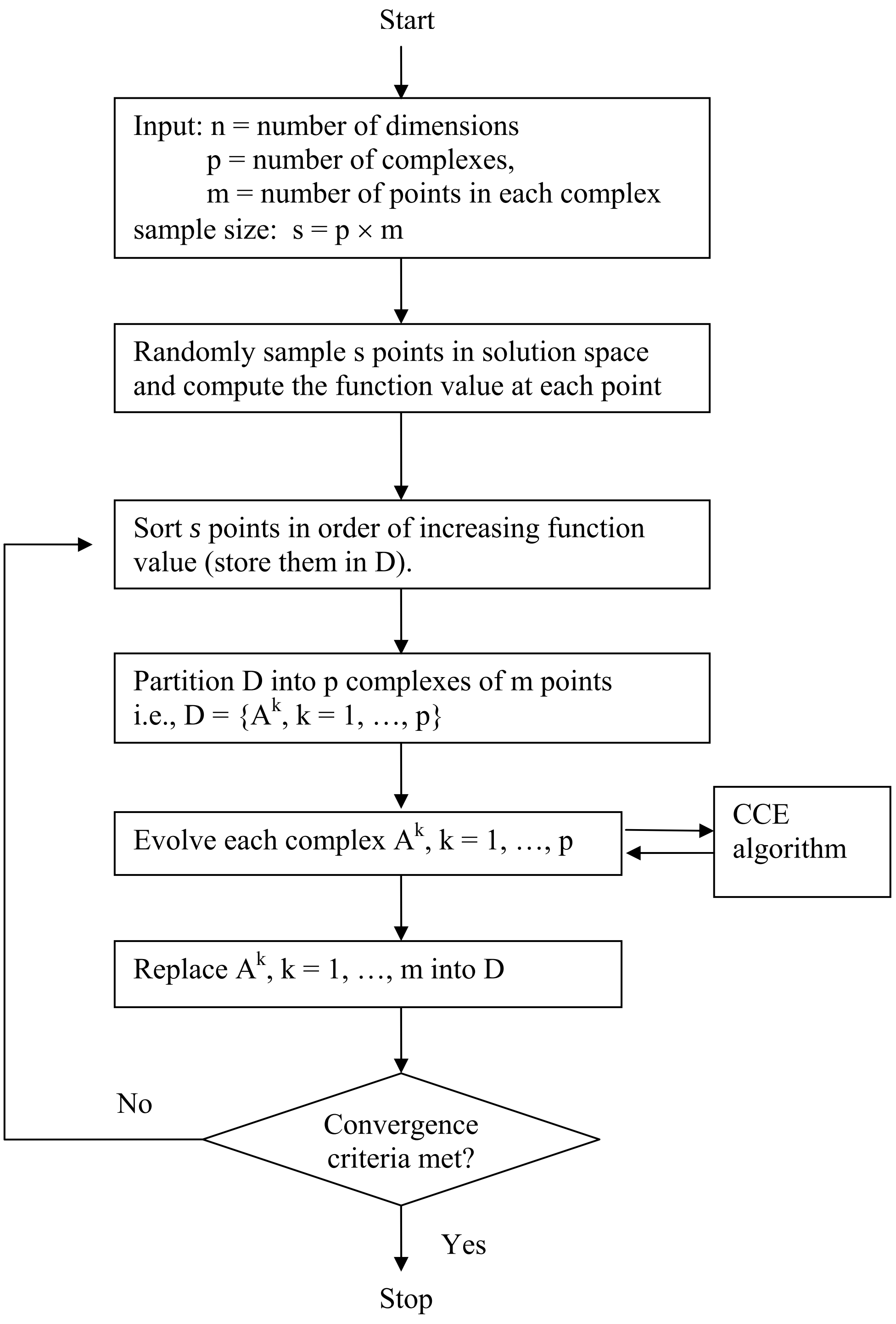


Figure 4.1 Schematic diagram of Evolutionary Algorithms (EAs)



Start
Input: n = number of dimensions
p = number of complexes,
m = number of points in each complex
sample size: s = p × m
Randomly sample s points in solution space and compute the function value at each point
Sort *s* points in order of increasing function value (store them in D).
Partition D into p complexes of m points i.e., D = {A^k, k = 1, …, p}
Evolve each complex A^k, k = 1, …, p
CCE algorithm
Replace A^k, k = 1, …, m into D
No
Convergence criteria met?
Yes
Stop

Figure 4.2 Search algorithm of Shuffled Complex Evolutions (SCE)

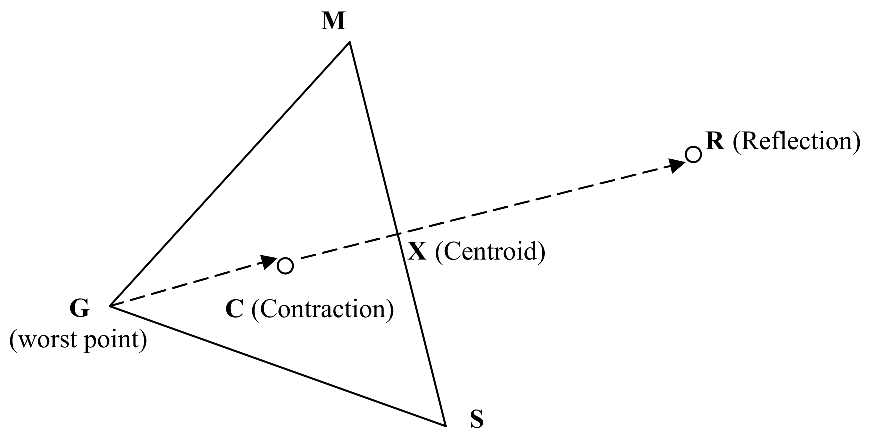


Figure 4.3 Basic processes in Competitive Complex Evolution (CCE): reflection and contraction

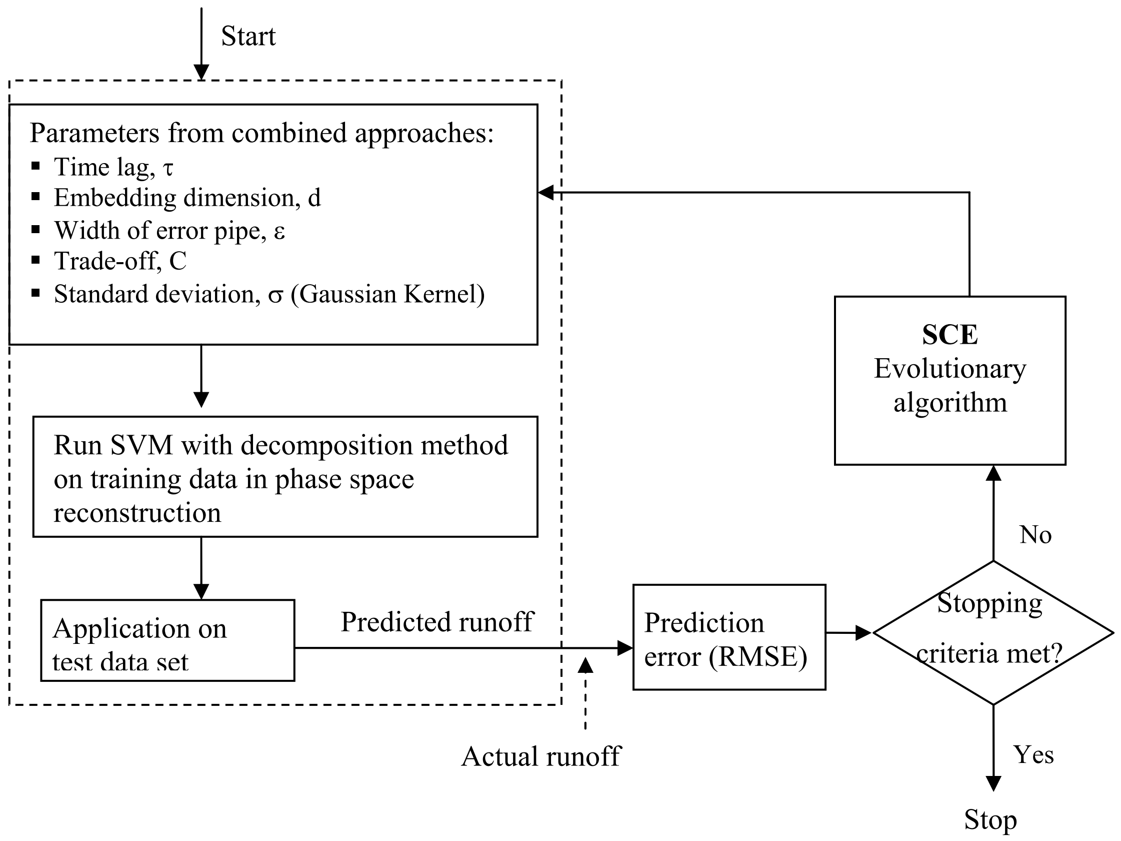
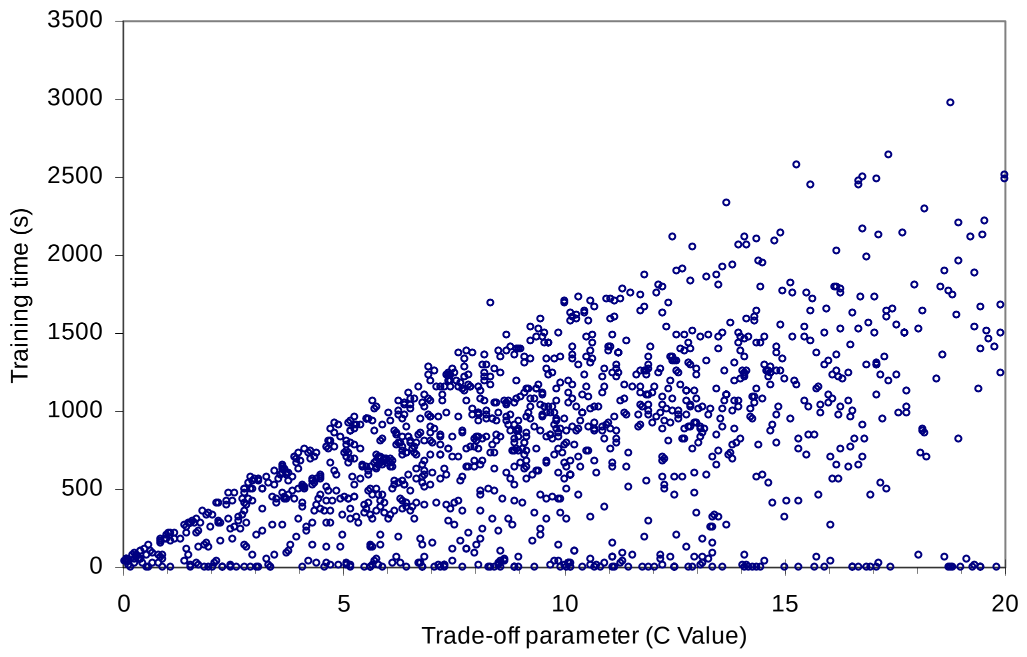
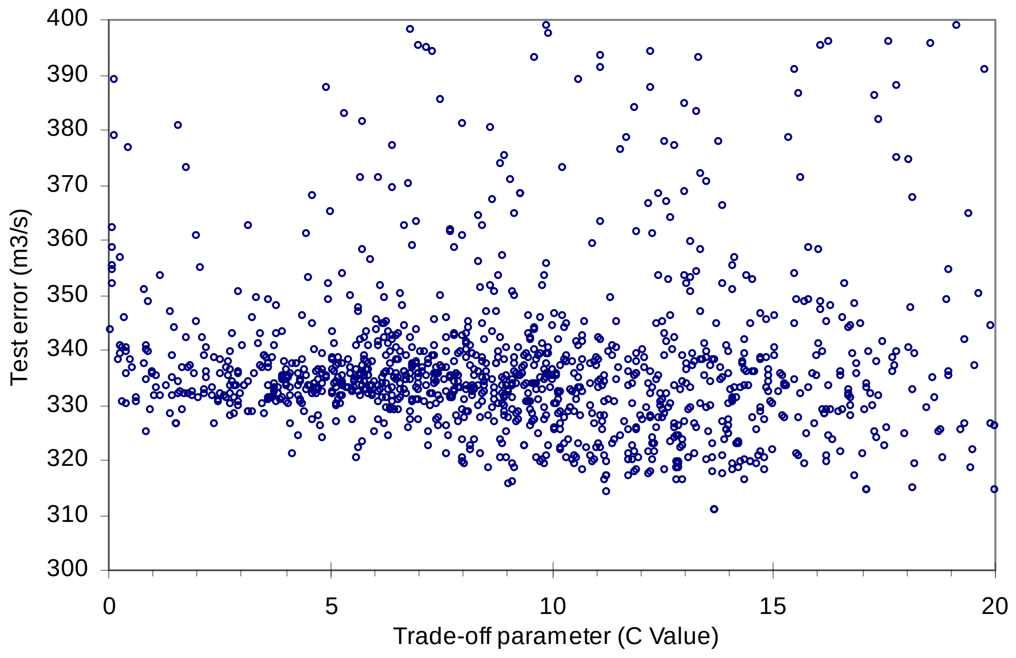

Start
Parameters from combined approaches:
Time lag, τ
Embedding dimension, d
Width of error pipe, ε
Trade-off, C
Standard deviation, σ (Gaussian Kernel)
Run SVM with decomposition method on training data in phase space reconstruction
Application on test data set
Predicted runoff
Actual runoff
Prediction error (RMSE)
Stopping criteria met?
No
Yes
Stop
SCE Evolutionary algorithm

Figure 4.4 Proposed algorithm of EC-SVM I

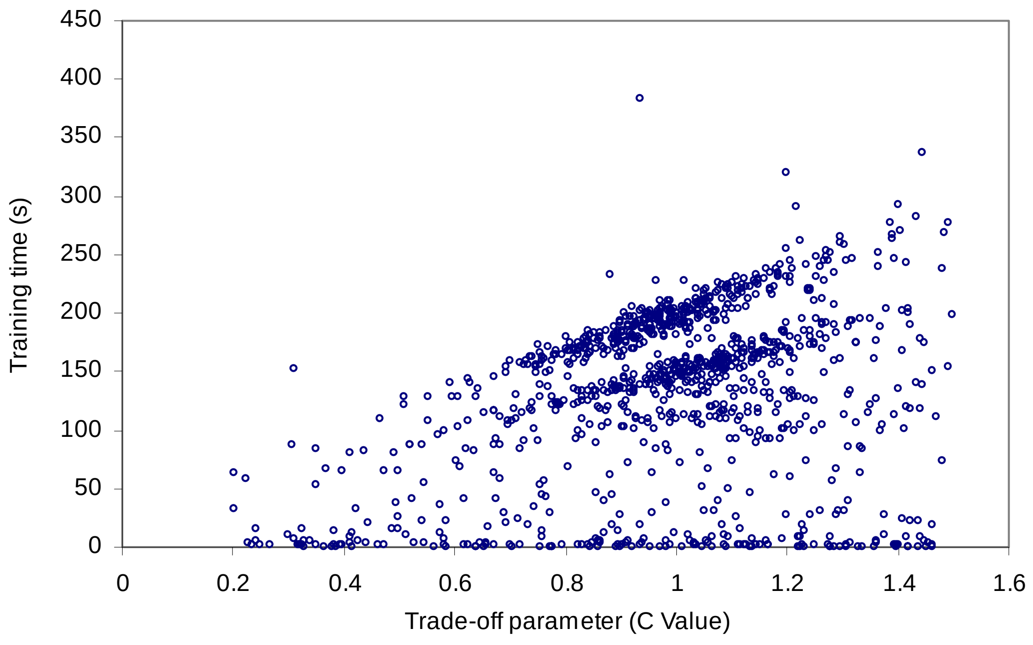


(a) Training time

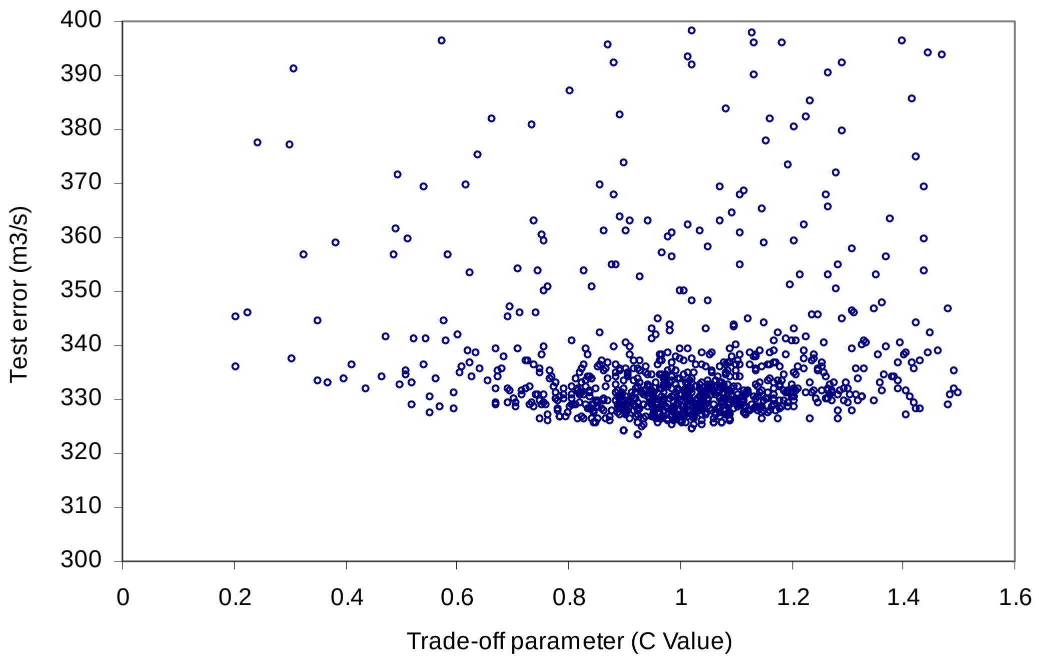


(b) Test error

Figure 4.5 Effect of varying C value on training time and test error: EC-SVM I



(a) Training time



(b) Test error

Figure 4.6 Effect of varying C value close to the output variable range B on training time and test error: EC-SVM I

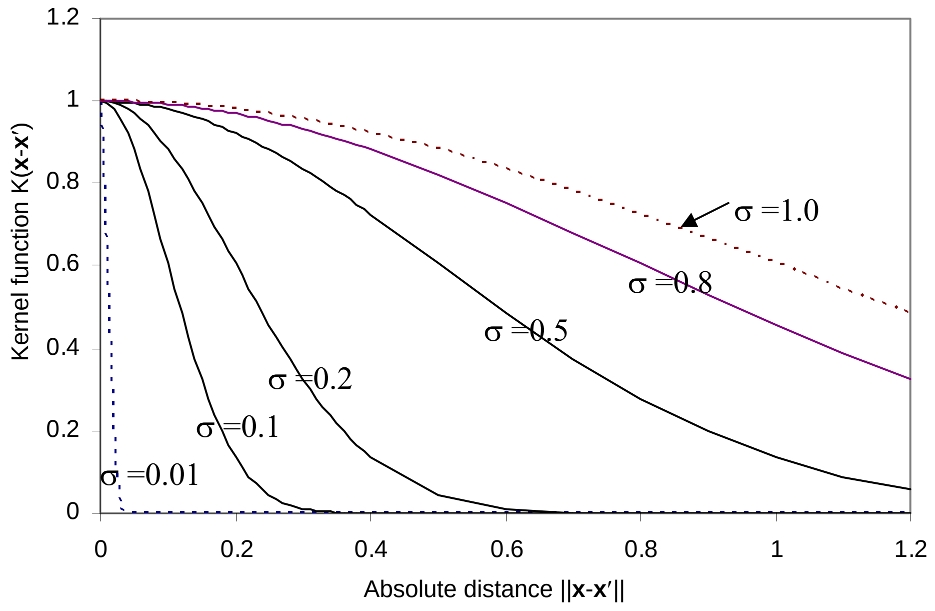


Figure 4.7 Sensitivity of varying Kernel widths σ

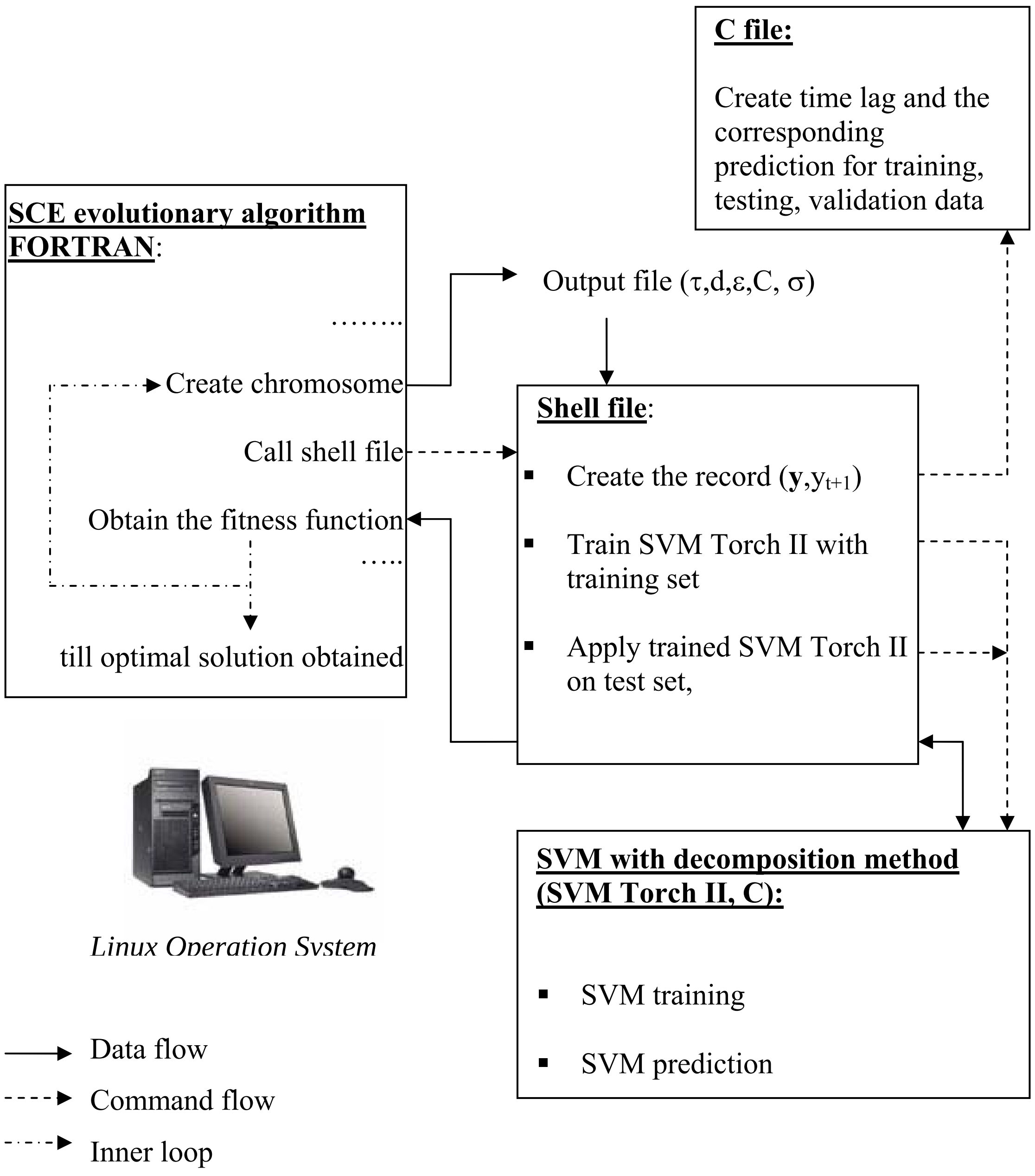

C file:
Create time lag and the corresponding prediction for training, testing, validation data
SCE evolutionary algorithm FORTRAN:
……..
Create chromosome
Call shell file
Obtain the fitness function
…..
till optimal solution obtained
Output file (τ,d,ε,C, σ)
Shell file:
Create the record (y,y_{t+1})
Train SVM Torch II with training set
Apply trained SVM Torch II on test set,
SVM with decomposition method (SVM Torch II, C):
SVM training
SVM prediction
Linux Operation System
Data flow
Command flow
Inner loop

Figure 4.8 Operational diagram of EC-SVM I

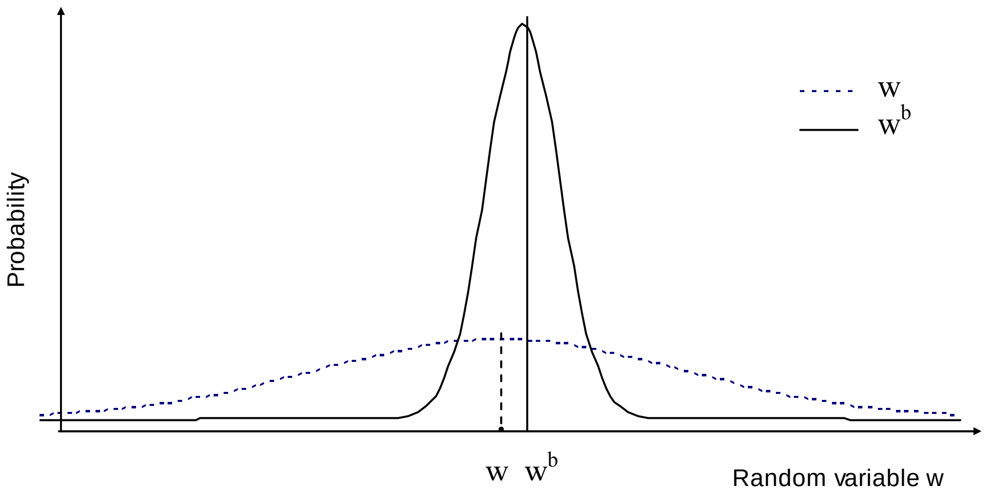
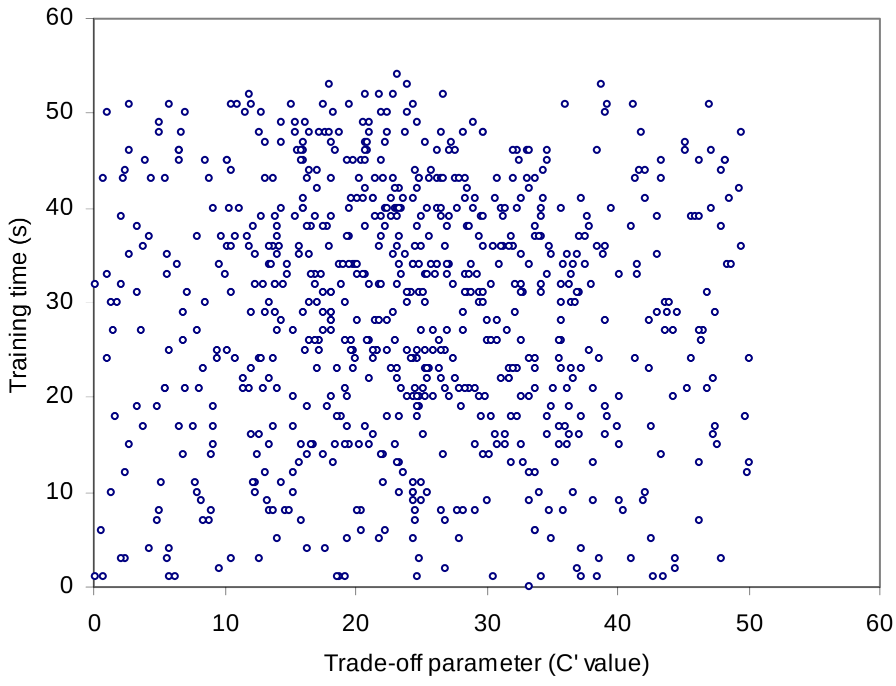
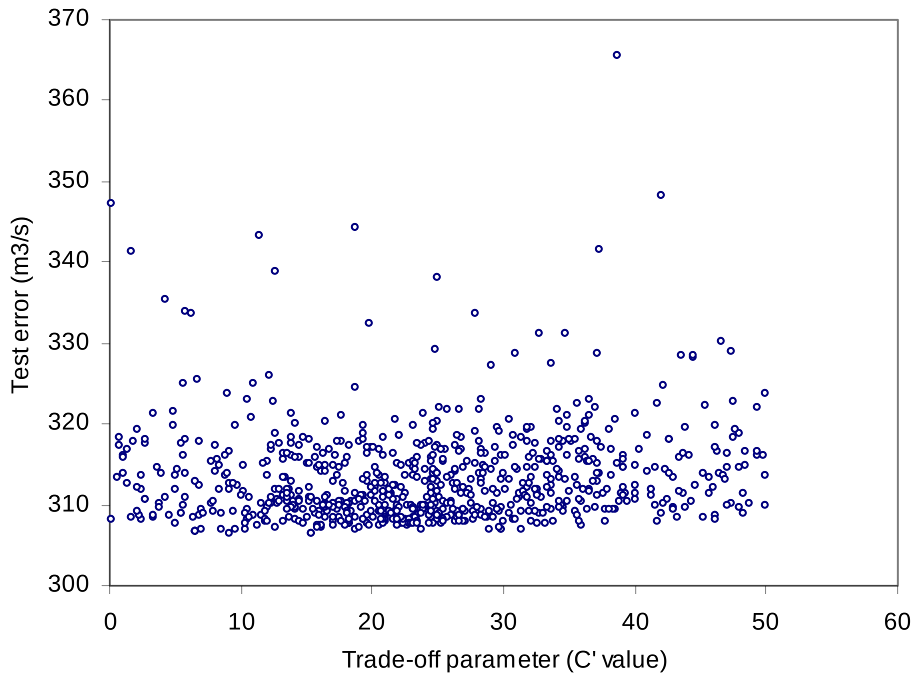


Figure 4.9Distinction between unbiased distribution with large variance estimation (w) and biased distribution with small variance estimation (w^b)

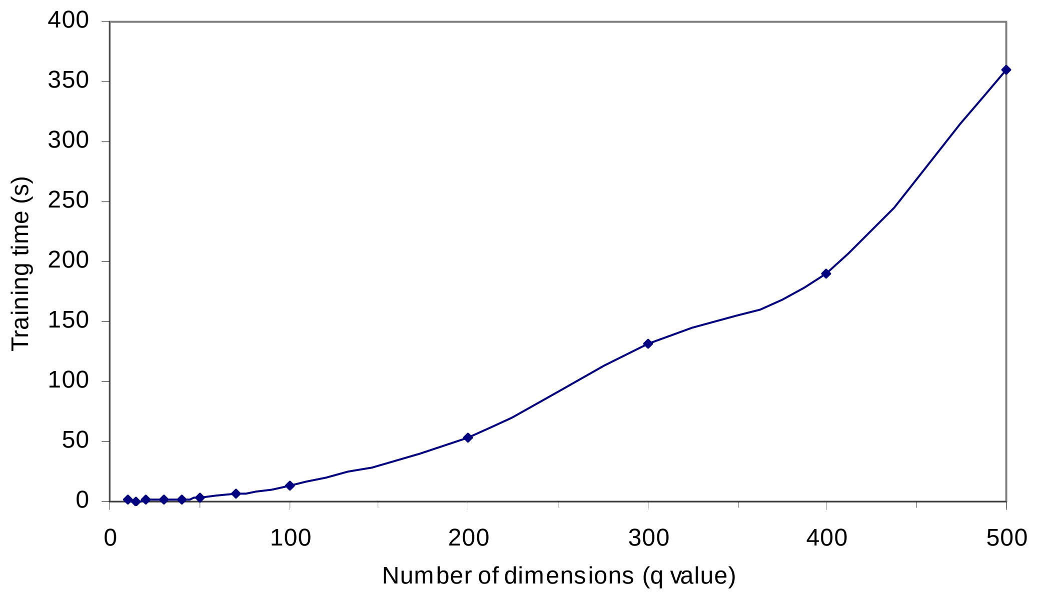


(a) Training time

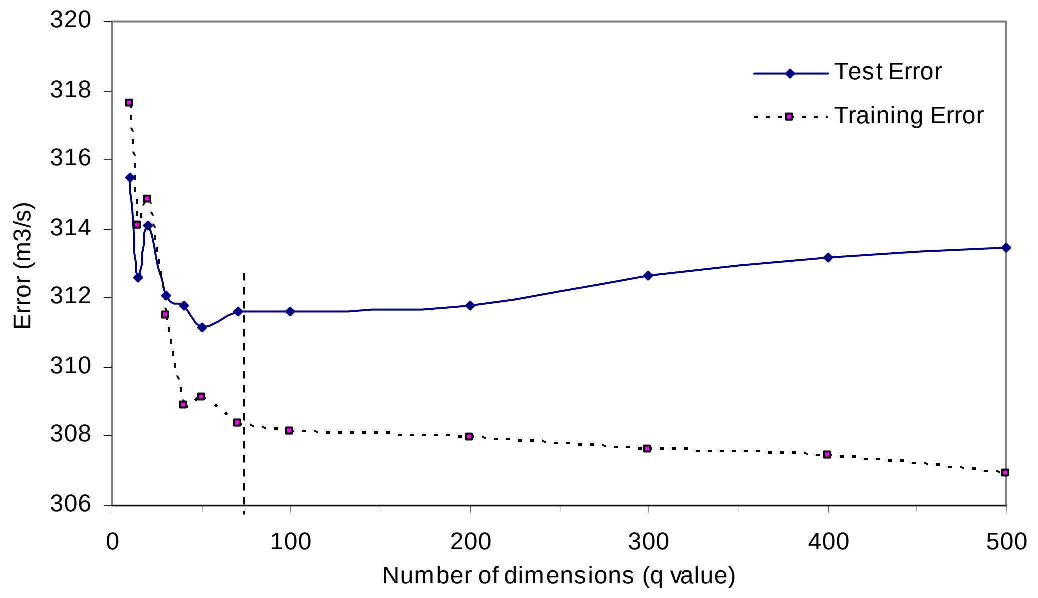


(b) Test error

Figure 4.10 Effect of varying C′ value on training time and test error: EC-SVM II

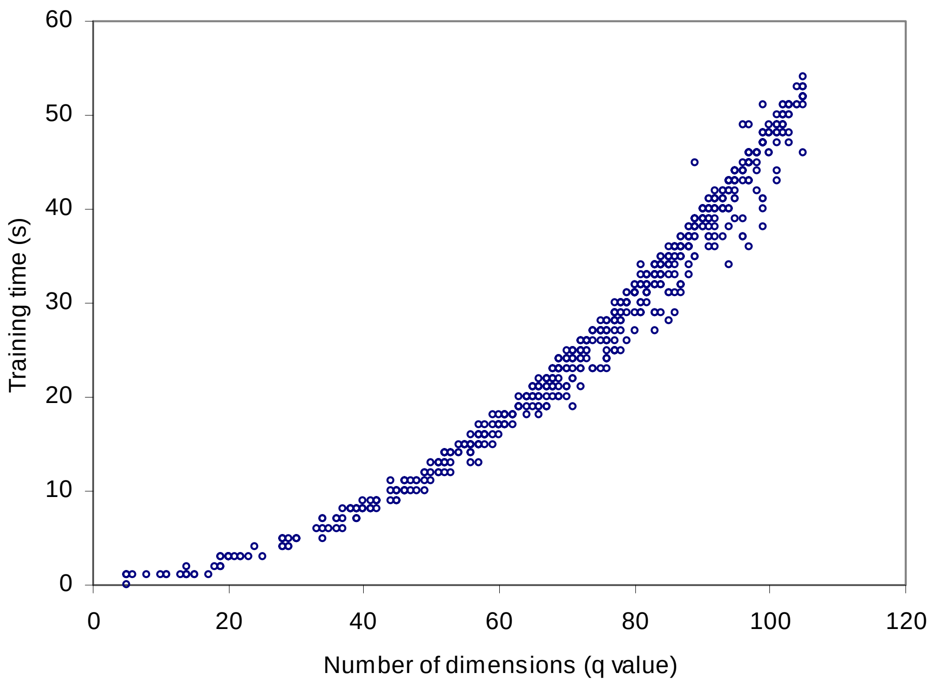


(a) Training time

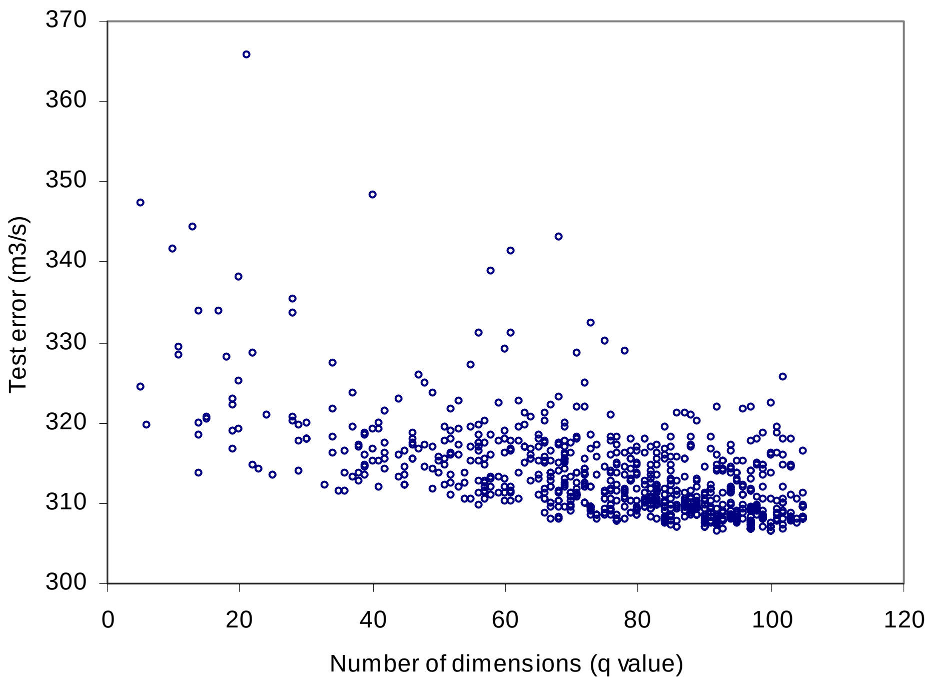


(b) Test error

Figure 4.11 Effect of varying number of dimensions (q) of approximated features on training time and test and training errors: EC-SVM II



(a) Training time



(b) Test error

Figure 4.12 Effect of number of dimensions (q) on training time and test error: EC-SVM II

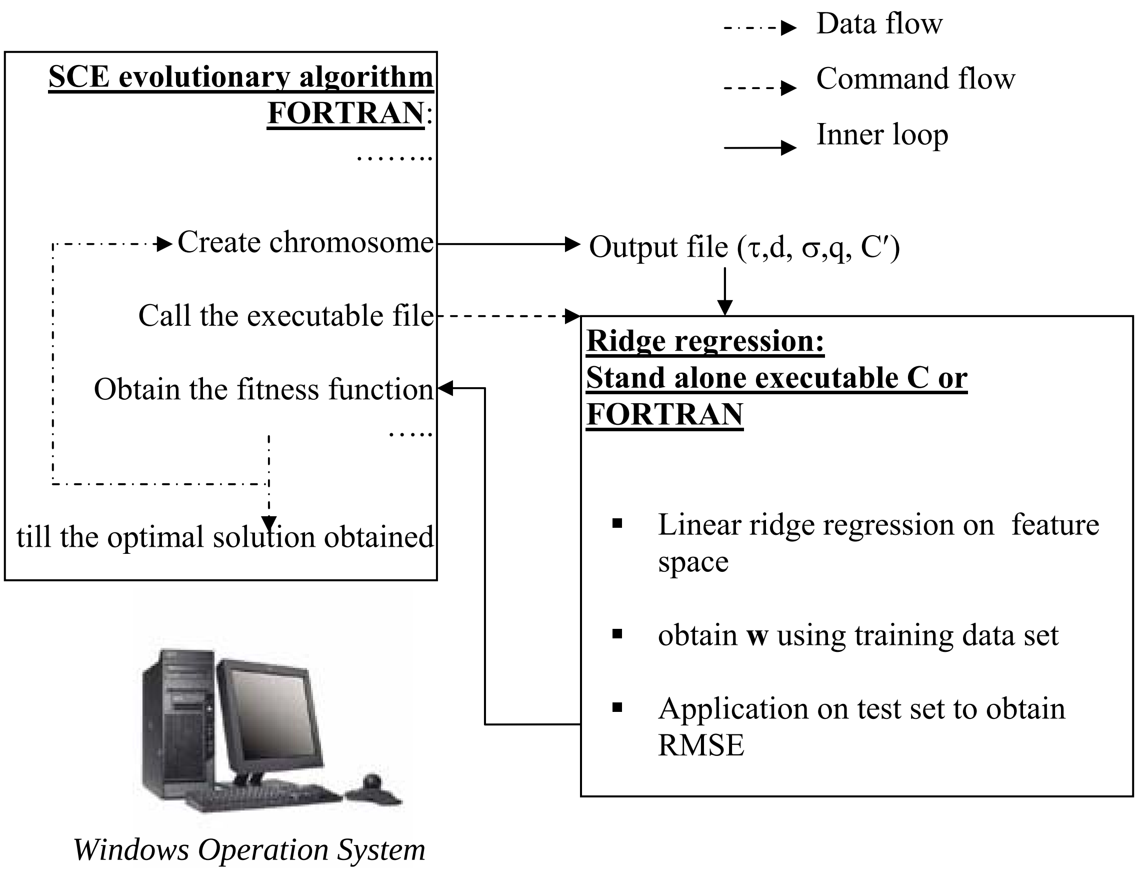

Data flow
Command flow
Inner loop
SCE evolutionary algorithm FORTRAN:
……..
Create chromosome
Call the executable file
Obtain the fitness function
…..
till the optimal solution obtained
Output file (τ,d, σ,q, C′)
Ridge regression:
Stand alone executable C or FORTRAN
Linear ridge regression on feature space
obtain w using training data set
Application on test set to obtain RMSE
Windows Operation System

Figure 4.13 Operational diagram of EC-SVM II

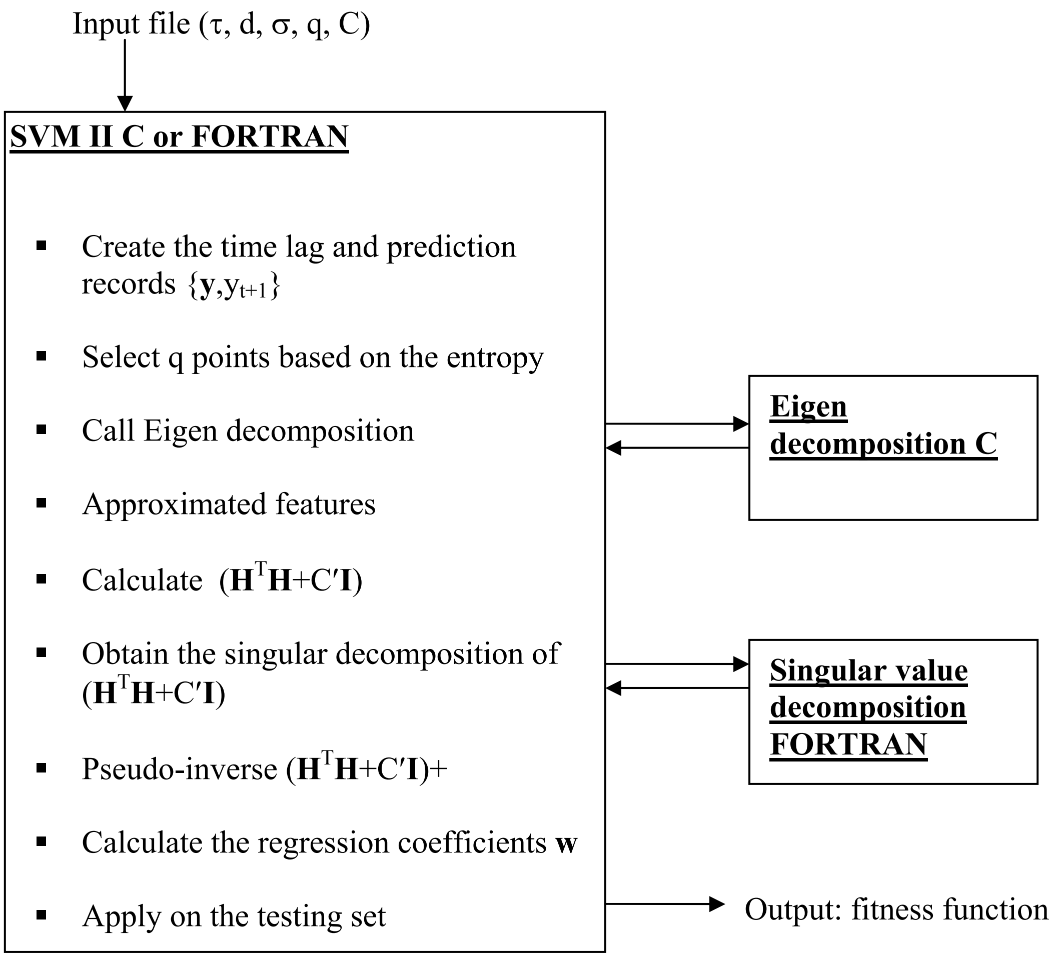

Input file (τ, d, σ, q, C)
SVM II C or FORTRAN
Create the time lag and prediction records $\{\mathbf{y}, y_{t+1}\}$
Select q points based on the entropy
Call Eigen decomposition
Approximated features
Calculate $(\mathbf{H}^T\mathbf{H}+C'\mathbf{I})$
Obtain the singular decomposition of $(\mathbf{H}^T\mathbf{H}+C'\mathbf{I})$
Pseudo-inverse $(\mathbf{H}^T\mathbf{H}+C'\mathbf{I})+$
Calculate the regression coefficients $\mathbf{w}$
Apply on the testing set
Eigen decomposition C
Singular value decomposition FORTRAN
Output: fitness function

Figure 4.14 Flow chart of the sub-modules in EC-SVM II

Table 4.1 Recommended SCE control parameters

Parameter	Description	Range	Recommended
p_{min}	No. of minimum required complexes in the population	$1 \le p_{min} \le p$	n**
m	No. of points in each complex	$m \ge 2$	2n+1*
q	No. of points in a sub-complex	$2 \le q \le m$	n+1*
α	No. of offsprings generated by each sub-complex	$\alpha \ge 1$	1*
β	No. of evolution steps taken by each complex before shuffling	$\beta \ge 1$	2n+1*

* Recommended by Duan et al., (1992)

** Recommended by Kuzcera (1997)

CHAPTER 5
APPLICATIONS OF EC-SVM APPROACHES

5.1 Introduction

In Chapter 4 the proposed EC-SVM was elaborated in detail. In this chapter, EC-SVM I and EC-SVM II will be applied to two real time series, the runoff of Tryggevælde catchment, Denmark, and the Mississippi river flow at Vicksburg, USA. Their performance, measured in terms of effectiveness and efficiency, are analysed and discussed.

5.2 Daily runoff time series

Two daily runoff time series are applied in this study: daily runoff time series of Tryggevælde catchment and the daily Mississippi river flow. The former is characterized with a small average flow of 1 m^3/s while the latter is of totally different orders of magnitude larger, viz.17,000m^3/s.

5.2.1 Tryggevælde catchment runoff

The daily runoff time series of the runoff Tryggevælde catchment covers the period from January 01, 1975 to December 31, 1993. Tryggevælde catchment is situated in the eastern part of sea land, north of the village Karise, Denmark. It is a small catchment with an area of 130km^2 (Fig. 5.1). The soils in the catchment are predominated by clay and a flow is very flashy, a typical runoff time series. The basic statistics of runoff time series of Tryggevælde catchment are: (1) mean flow = 0.98 m^3/s; (2) standard deviation = 1.37 m^3/s; (3) maximum flow = 11.07 m^3/s; and (4) minimum flow = 0.014 m^3/s.

A sample of the time series is shown in Fig. 5.2 in different time scales. It can be seen from the figure that there are distinct wet and dry periods in each year. The slow changes of runoff, low to high or high to low, indicate that the flow is highly correlated.

Time series is divided into three segments. The first, second and third segments are known as training set, test set, and validation set. Of the total of 19 years from 1975 to 1993, the first 15 years data are used as training set, the next 2 years as test set, and the last 2 years as validation set. Thus, the daily data from 1975 to 1991 are used to characterize the system in the standard approach and to optimize the embedding structure parameter set (τ, d) in EC-SVM approach. 15 years daily time series is equivalent to about 5500 records.

The Fourier transform shows a very broad band power spectrum while the correlation dimension shows a low dimension of around 2, as shown in Fig. 5.3. The time lag used for the correlation dimension calculation is obtained from the AMI method.

The time lag and embedding dimension resulting from AMI and FNN are shown in Fig. 5.4. The first minimum of the average mutual information occurs when time lag is about 12 while the minimum FNN occurs when the dimension is 5. Therefore, the embedding structure obtained from AMI and FNN is (τ=12, d=5).

5.2.2 Mississippi river flow

Mississippi river is one of the world's greatest river systems. It originates as a tiny outlet stream from Lake Itasca in northern Minnesota, draining through about 31 states of U.S.A. and finally reaches the Gulf of Mexico with a length of 3,705 kilometres. The area of the Mississippi river basin is around 3.2 million square kilometres. Average amount of water discharged to the Gulf is about 17,000 m^3/s. The whole

Mississippi river basin consists of six major sub-basins: Missouri, Upper Mississippi, Ohio, Tennessee, Arkansas-Red-White and Lower Mississippi.

The spring floodwaters cause very costly flooding. Although billions have been spent to reduce flood damages, recent floods have cost billions of dollars and significant loss of life. Further understanding of the river flow behaviours and patterns is one of the most fundamental issues for the understanding of the complex ecosystem and protection strategy.

The daily time series under consideration in this study is the Mississippi river flow measured at Vicksburg, Station No. 07289000 (Hydrologic Region 08 of USGS). The daily runoff time series is downloaded from the USGS. The station is located close to the entrance to the sea of Mississippi River, Fig. 5.5.

A sample of the time series is shown in Fig. 5.6 in different time scales. It can be seen that the flow pattern is smoother than that of Tryggevælde catchment runoff.

The basic statistics of daily Mississippi river flow time series are: (1) mean flow =18,456 m^3/s; (2) standard deviation=9,727m^3/s; (3) maximum flow = 52,103 m^3/s; and (4) minimum flow = 3,907 m^3/s. The daily runoff time series of the Mississippi rive flow covers the period from January 01, 1975 to December 31, 1993. The time series is divided into three segments as well; the first 15 years data are used for training set, the next 2 years for test set, and the last 2 years for validation set.

The Fourier transform also shows a very broad band power spectrum while the correlation dimension displays a low dimension of around 6 as shown in Fig. 5.7. The correlation dimension is slightly higher than that of the Tryggevælde catchment runoff.

The time lag and embedding dimension resulting from AMI and FNN are shown in Fig. 5.8. The first minimum of average mutual information occurs when time lag is

13 while the minimum FNN occurs when the dimension is 5. Therefore, the embedding structure obtained from AMI and FNN is (τ=13, d=5).

5.3 Applications of EC-SVM I on daily runoff time series

The proposed EC-SVM first reconstructs the phase space (following **C**haos analysis) and then optimizes both the **SVM** and embedding structure parameters simultaneously with an **E**volutionary algorithm. There are two techniques, attached to EC-SVM, to circumvent solving problem with large data record: one with decomposition method (EC-SVM I) while the other is with linear ridge regression (EC-SVM II). The applications of EC-VM I are demonstrated first in the following subsection while the applications of EC-SVM II are demonstrated in Section 5.4.

5.3.1 EC-SVM I on Tryggevælde catchment runoff

The results provided by other techniques will be briefly shown first. Only training and validation sets are required in these other techniques. Data from 1975 to 1991 serve for chaotic behaviour detection and phase space reconstruction while data from 1992 to 1993 serve for validation. Using (d =4, τ =12, k =5), the root mean square error (RMSE) for 1-lead day prediction used on the validation set is 0.647. Root mean square error (RMSE) is defined as:

$$RMSE = \sqrt{\textstyle\sum_{i=1}^{N} (y_i - \hat{y}_i)^2) / N} \qquad (5.1)$$

where y_i is the observed value and $\hat{y}_i$ is the predicted value.

Naive forecasting is a simple and yet often potentially effective time series forecasting technique particularly for short lead times. The forecast at time (t+1), for example, is assumed to be equal to the value observed at time t, i.e. $y_{t+1} = y_t$. Using naive forecasting, the RMSE for 1-lead day prediction, applied on the same validation set is 0.577. Thus, naive forecasting performs better than the traditional chaotic technique.

ARIMA (p, d, q) model is a mixed autoregressive-moving average model of order (p, q) on the d-th differences of the time series. The RMSE values for 1-lead day prediction, for validation set, resulting from the first order ARIMA (1,0,1) and ARIMA (1,1,1) are 0.535 and 0.543 respectively.

Phoon et al. (2002) and Liong et al. (2002) used the inverse approach on the same data set grouped exactly into three subsets as that done in this study and discussed in Section 5.2. The optimal (d, τ, k) set resulting from Phoon et al. (2002) and Liong et al. (2002) are (3,1,10) and (2,1,11) respectively. The RMSE values resulting from Phoon et al. (2002) and Liong et al. (2002), using their respective optimal parameters set, for the validation set are 0.540 m^3/s and 0.528m^3/s, respectively.

There are a total of 5 parameters in EC-SVM I, (τ, d, ε, σ, C). Once the ranges of these parameters are selected, EC-SVM I will search for the best choice of combination within these ranges. The parameter range set follows the suggestion given in Section 4.3; they are listed in Table 5.1. Since the embedding structure parameters obtained from AMI and FNN are (τ=13, d=5) as described in Section 5.3, it is sufficient to set τ at [1, 20] and d at [2, 20]. Data are normalized into [0, 1] range; runoff data Q under consideration is divided by the maximum runoff value, Q_{max}. It is sufficient to set ε at [0, 0.1] and σ at [0.1, 0.9]. There is no limit for C since it is a trade off between two items, empirical error and model complexity. There are different C value sets considered in this experiment; one is at [0.2, 1.5] while the other at [0.2, 20.0].

The stopping criteria for the search algorithm are:

(1) The population has converged to a pre-specified value, 0.0001, of the original parameter space,

(2) The relative change in the objective function within the last 5 shuffling loops has not changed more than a pre-specified percentage as 0.001 , and

(3) The total number of evaluations has exceeded a pre-defined value 1500.

The first difference of the time series is also considered in this study. The first difference of the time series is a common technique to obtain a more stationary time series. It is an attempt to further improve the prediction accuracy. The daily flow difference, or the first difference, dQ(t), is expressed as:

$$dQ(t) = Q(t+1) - Q(t) \tag{5.2}$$

The focus is now to predict the dQ(t) value. Similar to the Q time series, the phase space of dQ time series is first reconstructed followed by a dynamics reconstruction. The following function serves as the predictor of dQ(t):

$$dQ_{t+1} = f(dQ_t, dQ_{t-\tau}, \ldots, dQ_{t-(d-1)\tau}) \tag{5.3}$$

The prediction, dQ, is first conducted. Its value is then substituted into Q(t+1) = Q(t) + dQ(t).

The computational time and the prediction accuracy of EC-SVM I on Tryggevælde catchment runoff are shown in Table 5.2 for both Q and dQ time series. The results shown are based on the program running on LINUX Pentium II 333MHz. The optimal parameter set is shown in Table 5.3. The minimum validation RMSE error is 0.521 m^3/s when dQ time series is used and the range of C is set at [0.2 20].

The training time increases as the upper bound of C range increases as shown in Fig. 5.9. It can be seen that the training time for dQ time series is shorter than that for Q time series. As the upper bound of C is increased to 50, the training time becomes very long due to numerical instability. The training time can be longer than one week for an iteration number of about 1000.

Table 5.4 shows the results from Q and dQ time series, with various techniques, of Tryggevælde catchment runoff. The result shows that EC-SVM I has a better performance than other techniques as shown in Table 5.4. EC-SVM I scheme on Q time series yields 19.3% improvement over the standard chaotic techniques. EC-SVM I achieves further prediction improvement by the analysis conducted on dQ time series. EC-SVM I on dQ time series provides the highest prediction accuracy with RMSE value of 0.521m^3/s.

The convergence of EC-SVM I on Tryggevælde catchment runoff is shown in Fig. 5.10. Figure 5.11 shows the hydrograph comparison between EC-SVM I simulated (based on dQ time series) and that observed.

5.3.2 EC-SVM I on Mississippi river flow

In this section the daily flow time series of Mississippi river at Vicksburg, Station No. 07289000 (Hydrological Region 08 of USGS), is considered. Similar to the approach shown in Section 5.3.1, for traditional chaotic technique data are divided into two parts: (1) 1975 to 1991 for phase space reconstruction; and (2) 1992 to 1993 for forecasting. With ($d = 6$, $\tau = 13$, $k = 7$) the resulting RMSE from the forecasting set is 1738.95m^3/s.

Liong et al. (2002) also analysed the same set of Mississippi river data and divided them into training (1975-1989), test (1990 – 1991) and validation (1992- 1993) sets. Liong et al. (2002) reported a RMSE value of 356.89m^3/s, with an optimal parameter set ($d = 2$, $\tau = 1$, $k = 5$), for the validation set (1992 – 1993). Using naive forecasting, the prediction yields a RMSE of 608.70m^3/s.

The computational time and the prediction accuracy of EC-SVM I on daily Mississippi rive flow are shown in Table 5.5 for both Q and dQ time series. The resulting optimal parameter sets are shown in Table 5.6. The minimum validation RMSE error is 302.4 m^3/s when dQ time series is used and the range of C is set at [0.2 1.5].

The training time for Mississippi river flow is longer than that for Tryggevælde catchment runoff time series. The results shown are based on the program running on LINUX Pentium II 333MHz. Similarly the training time increases as the upper bound of C range increases as shown in Fig. 5.12. It can be seen that the training time for dQ time series is less than that for Q time series. The training time increases significantly as the upper bound of C range increases. The training time increases to 300 hours as C values' upper bound increases. The same is observed for dQ time series; however, it is much less than its counterpart from Q time series.

Table 5.7 shows prediction accuracies resulting from Q and dQ time series, with various techniques, of Mississippi river flow. EC-SVM I on dQ time series yields the highest prediction accuracy with RMSE of 302.40m^3/s. It can be seen that the EC-SVM I approach on Q time series yields a significant improvement, 82.3%, over the standard chaotic approach; 49.4% improvement over Naive approach; 29.2% improvement over ARIMA (1, 0, 1) model; and 13.7% improvement over inverse approach with local model. EC-SVM I applied on dQ time series yields a better prediction performance than that of Q time series.

Figure 5.13 shows the evolutional convergence of EC-SVM I on Mississippi river flow data. Figure 5.14 shows the hydrograph comparison between EC-SVM I simulated (with dQ time series analysis) and that observed.

5.3.3 Summary

The proposed EC-SVM I, a forecasting tool with SVM operating in the Chaos inspired phase space and optimised with an Evolutionary algorithm, has been applied to two real daily flow time series, runoff of Tryggevælde catchment and Mississippi river flow. A recently developed decomposition method was seen to be suitable in chaos time series analysis since the method is able to deal with large data records.

The study shows that the proposed EC-SVM I provides more accurate prediction than the traditional chaos technique and naive forecasting. For Tryggevælde catchment runoff, RMSE is decreased from 0.647m^3/s to 0.521 m^3/s; for Mississippi river flow, the RMSE is significantly reduced from 1738.95m^3/s to 302.40m^3/s.

The study further suggests to apply the daily flow differences time series instead of the flow time series since the computational speed is significantly much faster.

The application shows that the search can find the optimal solution within less than 1000 iterations. It is very efficient to use the evolutionary algorithm as a search engine to calibrate the parameters. The training time increases as C increases. As the upper bound of C ranges to 20, the training time for Mississippi river flow is over 300 hours. The training time for Mississippi river flow time series is longer than that of Tryggevælde catchment runoff.

5.4 Applications of EC-SVM II on daily runoff time series

The decomposition method is used in EC-SVM I to deal with large scale data sets. Since it is an iterative algorithm, the computational time can be quite long. To overcome the uncertainty with long computational simulation time, a linear ridge regression method is therefore considered and included in EC-SVM II.

The applications of EC-VM II are demonstrated in the following subsection for runoff of Tryggevaelde catchment and Mississippi river flow. The stopping criteria of SCE are set as that given in Section 5.3. The range of C' value in EC-SVM II is broader than that of C in EC-SVM I. Table 5.8 shows the range of the parameters in EC-SVM II.

5.4.1 EC-SVM II on Tryggevælde catchment runoff

The computational time and the prediction accuracy of EC-SVM II on Tryggevælde catchment runoff, on both Pentium II 333MHz and Pentium IV 2.4GHz, are shown in Table 5.9 for both Q and dQ time series. The optimal parameter set is shown in Table 5.10. The minimum validation RMSE error is 0.500m^3/s when dQ time series is used.

The difference between the training times of EC-SVM II applied to Q time series and dQ time series of Tryggevælde catchment runoff are not as large as that of EC-SVM I. EC-SVM II for dQ time series and Q time series is, however, significantly faster than their counterparts with EC-SVM I. For Q time series, EC-SVM II is 2 times faster than EC-SVM I with C range set at [0.2 1.5]; and 5 times faster than EC-SVM I with C range set at [0.2 20.0]. The training time of EC-SVM II for Q time series is only 5 hours and 25 minutes, for 824 iterations on PII 33MHz; only 1 hour and 24 minutes on P4 2.4GHZ. The training time of EC-SVM II for dQ time series is slightly longer than that with Q time series since it performs more iterations, 1070 iterations.

Table 5.11 shows the results from Q and dQ time series, with various techniques, of Tryggevælde catchment runoff. The result shows that EC-SVM II for dQ time series has better prediction accuracy than other techniques as shown in Table 5.11. The prediction accuracy of EC-SVM II is better than that of EC-SVM I.

EC-SVM II scheme on Q time series yields 22.6% improvement over the standard chaotic techniques applied on Q time series. EC-SVM II achieves a slightly higher improvement when it is applied on dQ time series. EC-SVM II on dQ time series provides the highest prediction accuracy with RMSE value of 0.500m^3/s. Figure

5.15 shows the scatter plot of EC-SVM II on dQ time series analysis of Tryggevælde catchment runoff.

5.4.2 EC-SVM II on Mississippi river flow

The computational time and the prediction accuracy of EC-SVM II on Mississippi river flow, both on Pentium II 333MHz and Pentium IV 2.4GHz, are shown in Table 5.12 for both Q and dQ time series. The optimal parameter set is shown in Table 5.13. The minimum validation RMSE error is 300.71m^3/s when dQ time series is applied.

The difference between the training times of EC-SVM II applied to Q time series and dQ time series of Mississippi river flow, is not as large as that of EC-SVM I. EC-SVM II for dQ time series and Q time series are significantly faster than those of EC-SVM I. For Q time series, EC-SVM II is 3 times faster than EC-SVM I with C range set at [0.2 1.5], and 35 times faster than EC-SVM I with C range set at [0.2 20.0]. The training time of EC-SVM II with Q time series is 8 hours and 40 minutes for 1214 iterations, on PII 333 Mhz; only 2 hours 14 minutes on P4 2.4 GHz PCs. The training time of EC-SVM II on Mississippi river flow with dQ time series is shorter than that with Q time series; dQ time series requires only 762 iterations.

Table 5.14 shows prediction accuracies resulting from both Q and dQ time series, with various techniques, of Mississippi river flow. EC-SVM II on dQ time series yields the highest prediction accuracy with RMSE of 300.71m^3/s. It can be seen that the EC-SVM II approach on Q time series yields a significant improvement, 81.6%, over the standard chaotic approach; 47.3% improvement over Naive approach; 26.3% improvement over ARIMA (1, 0, 1) model; and 10.2% improvement over inverse approach with local model. EC-SVM II applied on dQ time series yields a better prediction performance than that of Q time series. Figure 5.16 shows the scatter plot of dQ time series analysis with EC-SVM II.

5.5 Comparison between EC-SVM I and EC-SVM II

The EC-SVM novel approaches proposed in this study have shown a better, in some cases significantly better, performance over other techniques, such as standard chaotic techniques using AMI, FNN and KNN method, Naïve forecasting, ARIMA model, or inverse approach with KNN method.

In the following subsections, comparison between EC-SVM I and EC-SVM II in detail is conducted.

5.5.1 Accuracy

The prediction accuracies of EC-SVM I and EC-SVM II with either Q or dQ time series of Tryggevælde catchment runoff and Mississippi river flow are shown in Table 5.15. The prediction accuracy of EC-SVM I listed in Table 5.15 is the higher one among C range set at [0.2 1.5] and at [0.2 20]. Figure 5.17 shows results from both Q and dQ time series.

EC-SVM II provides better prediction accuracy on Tryggevælde catchment runoff for both Q and dQ time series; however, its performance for the Mississippi river flow is better than EC-SVM I only for the dQ time series. The study shows that analysis with EC-SVM II on dQ time series yields the highest prediction accuracy.

5.5.2 Computational time

The computational times required by EC-SVM I and EC-SVM II, when they are applied to Tryggevælde catchment runoff and Mississippi river flow, are listed in Table 5.16. The EC-SVM I results listed in Table 5.16 is with C range set at [0.2 1.5], faster than that of C range set as [0.2 20]. Figure 5.18 illustrated them for both Q and dQ time series.

The computational time of EC-SVM II on all cases is shorter than that of EC-SVM I. EC-SVM II is around 2 times faster than that of EC-SVM I with Q time series of Tryggevælde catchment runoff and dQ time series of Mississippi river flow; 7% faster with dQ time series of Tryggevælde catchment runoff; and about 3 times faster with Q time series of Mississippi river flow.

5.5.3 Overall performances

Figure 5.19(a) shows detailed results of prediction accuracy, RMSE, on the validation data set and training time of EC-SVM I, with C range set at [0.2 1.5], for dQ time series of Tryggevæld catchment runoff. Similarly, Fig. 5.19(b) shows that of EC-SVM II. Results from EC-SVM I are more scattered that those of EC-SVM II. The training time and the prediction accuracy of EC-SVM I vary more than their counterparts in EC-SVM II. The training times of EC-SVM I for different iterations vary from about 0 to 140 seconds while EC-SVM II from about 0 to 60 seconds. The prediction accuracy varies from 0.5 to 0.75m^3/s for EC-SVM I and from 0.49 to 0.58m^3/s for EC-SVM II.

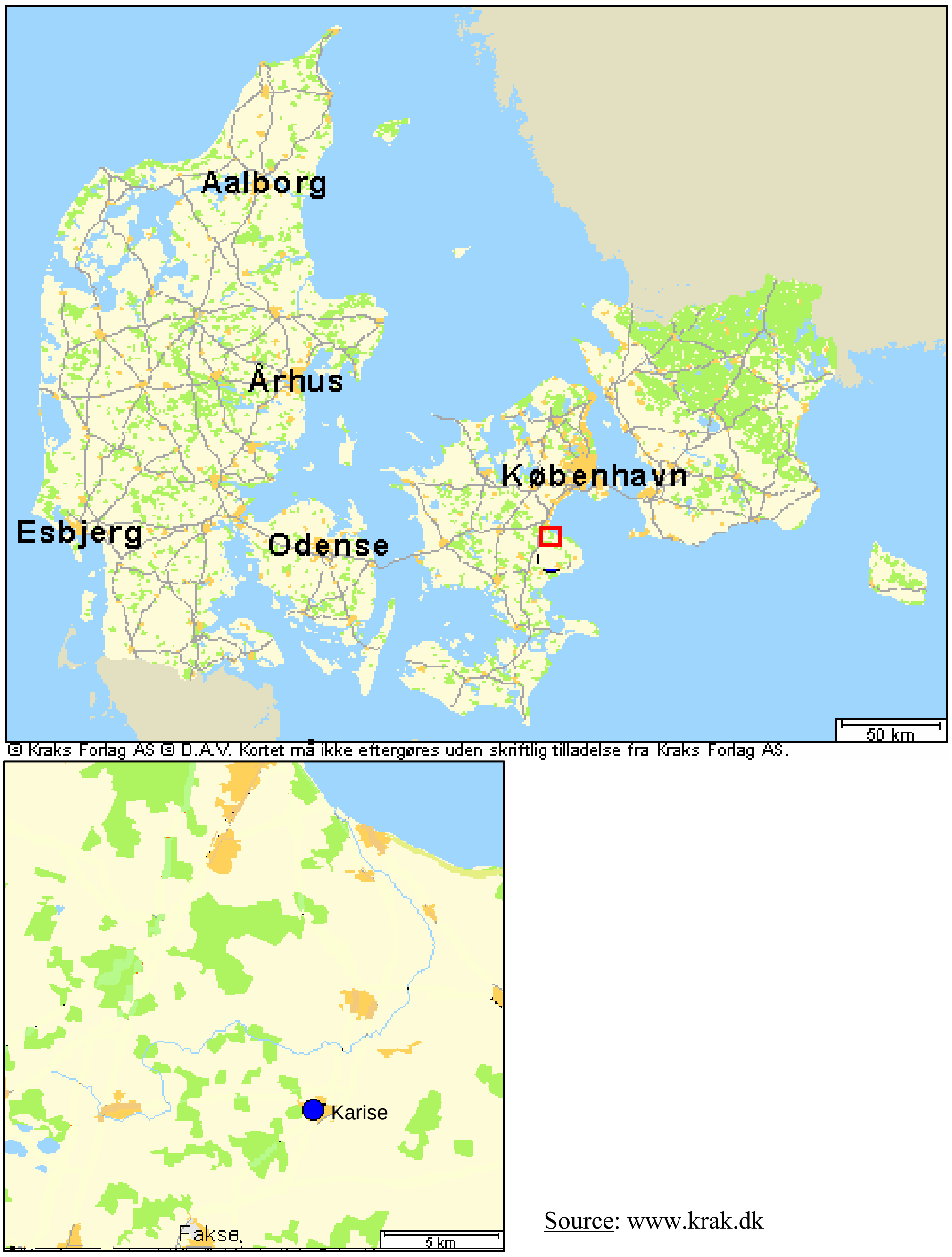
Figure 5.20(a) shows detailed results of prediction accuracy and training time for the validation data set of EC-SVM I, with C range at [0.2 1.5], applied to dQ time series of Mississippi river flow. Figure 5.20(b) shows results from EC-SVM II. Results from EC-SVM I are more scattered that those of EC-SVM II. The training times for various iterations vary from about 0 to 500 seconds for EC-SVM I while EC-SVM II from about 0 to 60 seconds. The prediction accuracy varies from 300 to 490 m^3/s for EC-SVM I while its counterpart, EC-SVM II from 297 to 340 m^3/s. Better performance of EC-SVM II is clearly demonstrated for Mississippi river flow when dQ time series is used.

5.6 Summary

The applications of the proposed EC-SVM I and EC-SVM II on Tryggevæld catchment runoff time series and Mississippi river flow time series are demonstrated in this chapter.

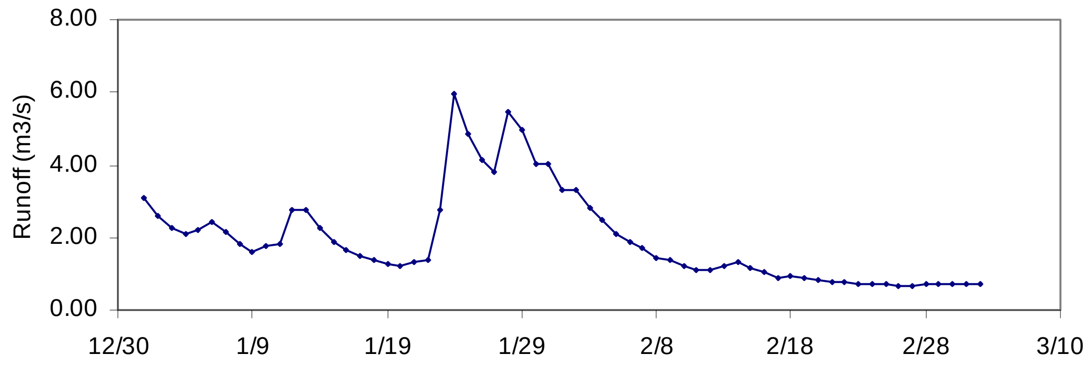
High prediction accuracies are obtained from the proposed EC-SVM I and EC-SVM II methods for both data sets. The novel approaches provide better prediction accuracy than traditional chaotic techniques, e.g. Naïve forecasting, ARIMA or inverse approach with KNN method. The proposed approach applies SVM on data in the phase space reconstruction and couples SVM with an evolutionary algorithm to calibrate both the embedding structure parameters and the SVM parameters.

Between the two schemes, EC-SVM I and EC-SVM II, the performance of the latter is more effective (higher accuracy degree) and more efficient (shorter computational time). Regarding the data set, it is suggested to consider its first difference, dQ time series, instead of the original Q time series.

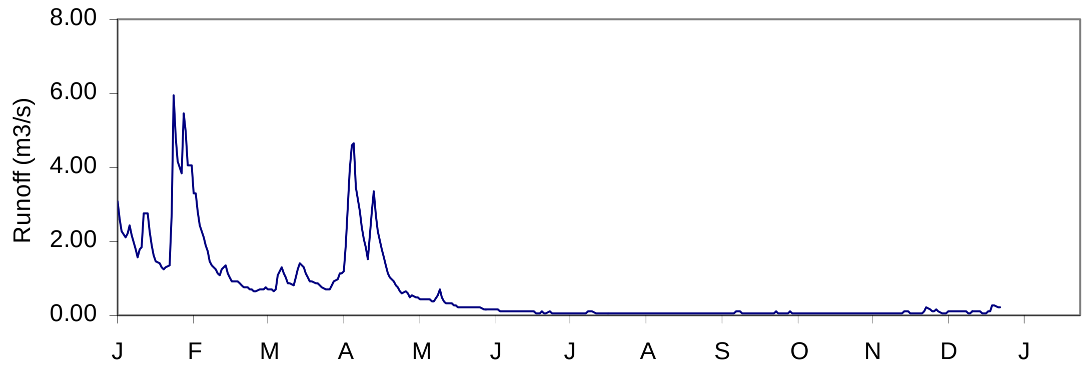

Aalborg
Århus
København
Esbjerg
Odense
50 km
© Kraks Forlag AS © D.A.V. Kortet må ikke eftergøres uden skriftlig tilladelse fra Kraks Forlag AS.
Karise
Fakse
5 km

Source: www.krak.dk

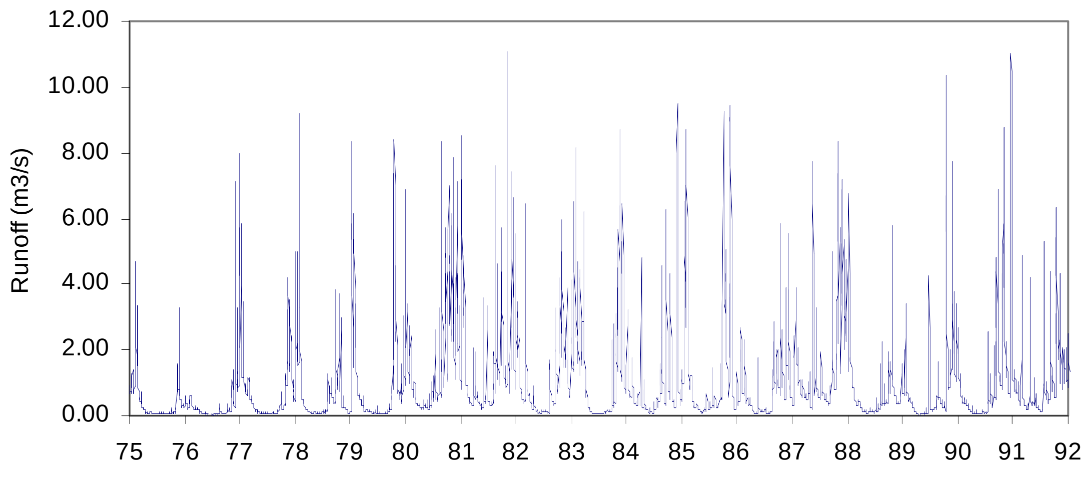
Figure 5.1 Location of Tryggevælde catchment, Denmark



(a) Daily scale

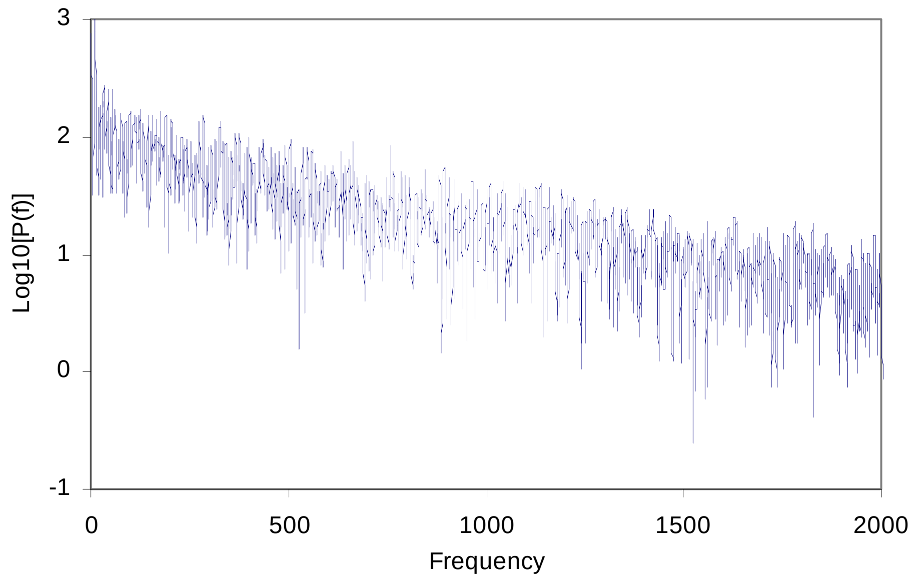


(b) Monthly scale

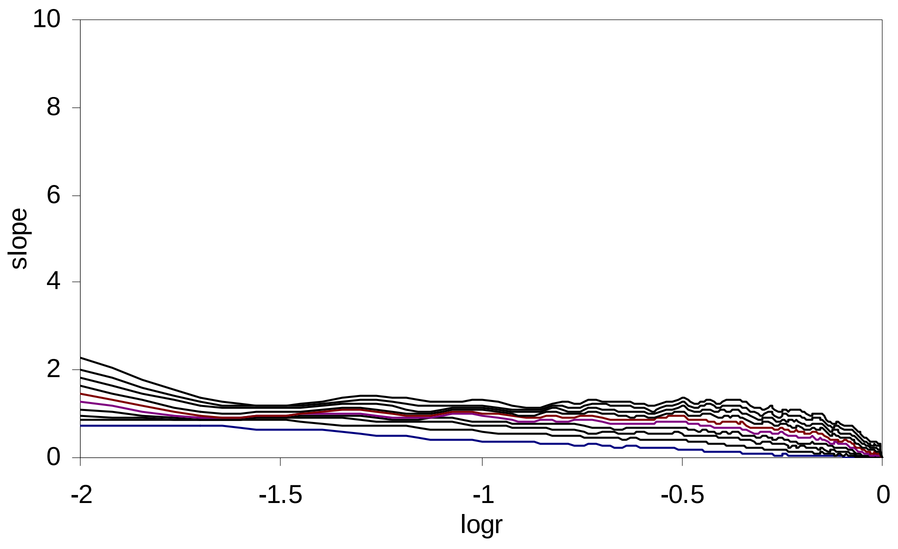


(c) Yearly scale

Figure 5.2 Daily runoff time series of Tryggevælde catchment plotted in different time scales

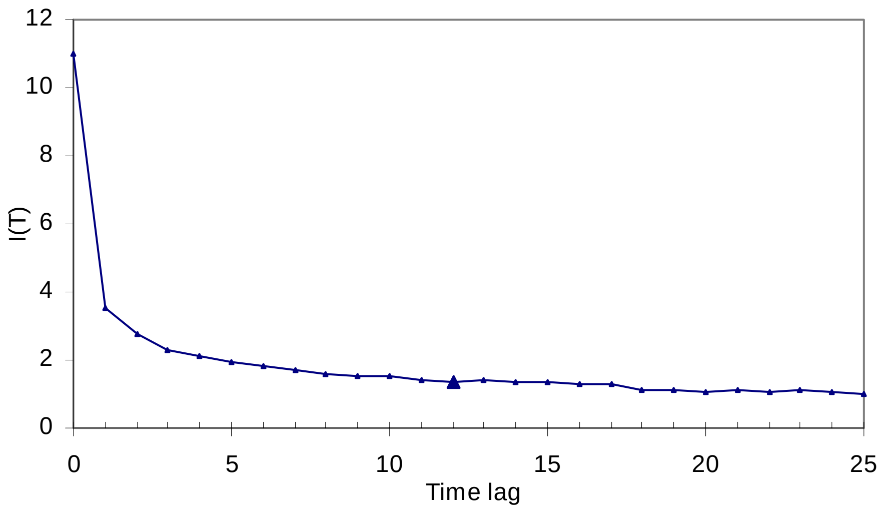


(a) Fourier transform

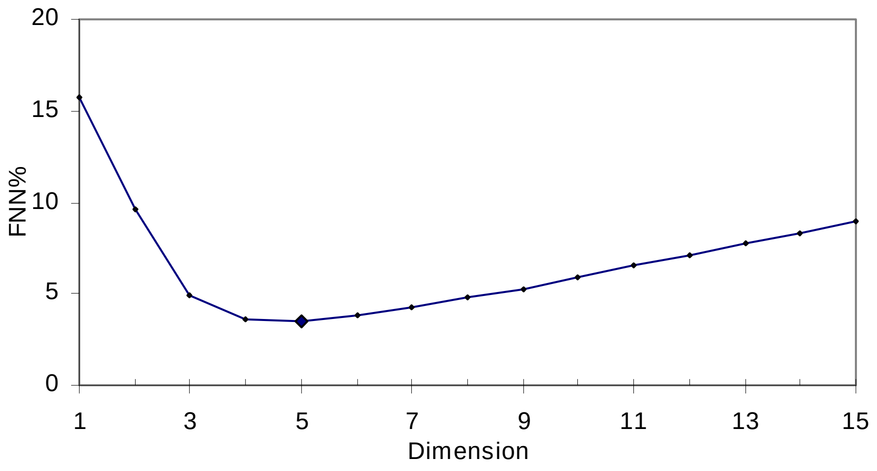


(b) Correlation dimension

Figure 5.3 Fourier transform and correlation dimension of daily Tryggevælde catchment runoff time series



(a) Average Mutual Information



(b) False Nearest Neighbours

Figure 5.4 Determination of time lag and embedding dimension: Tryggevælde catchment runoff time series

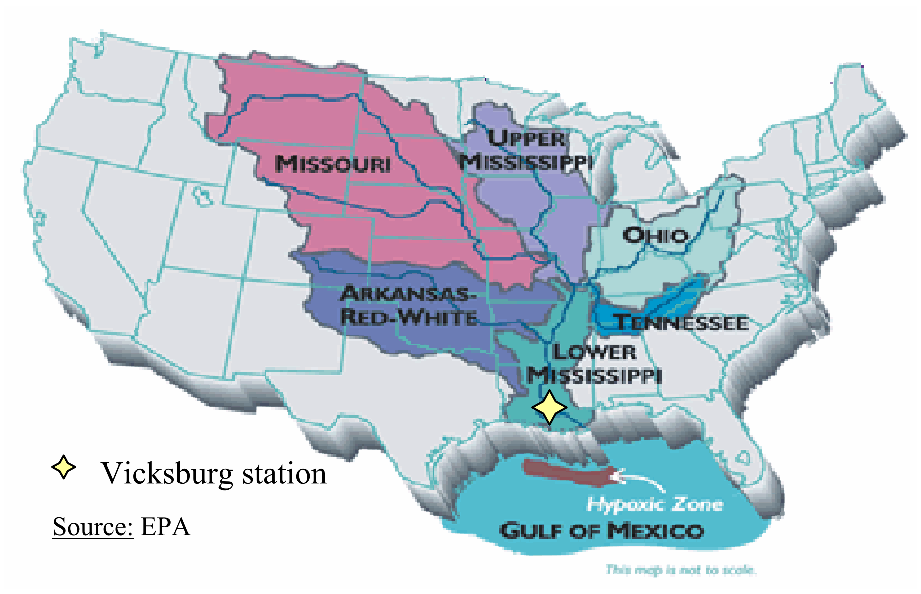
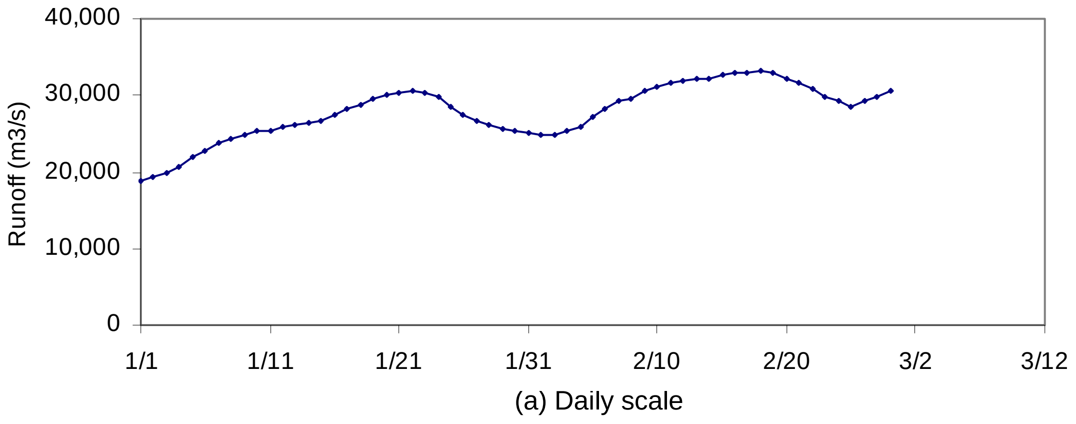
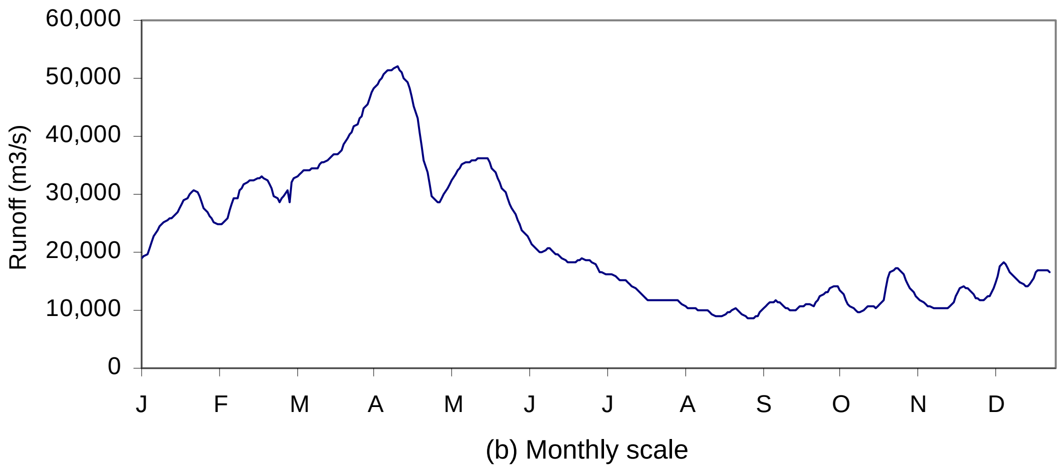


Figure 5.5 Location of Mississippi river, U.S.A. and runoff gauging station





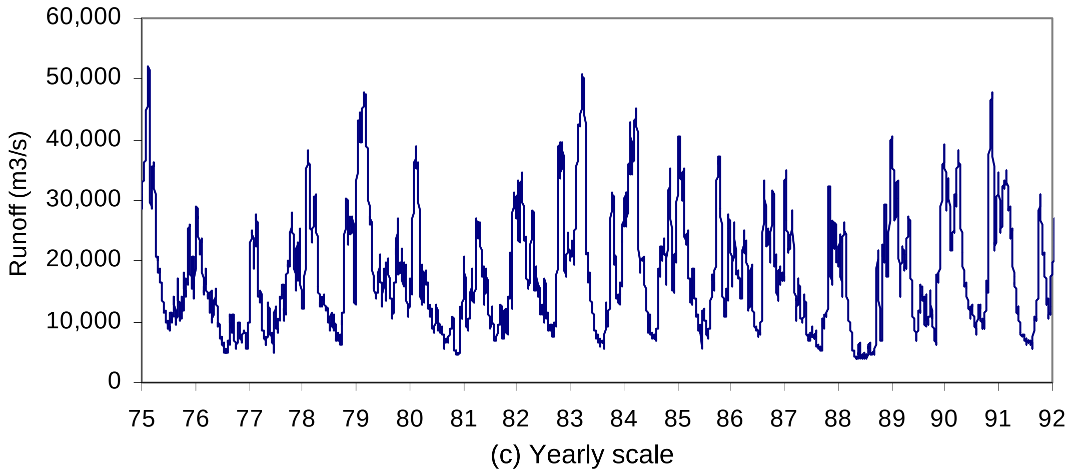
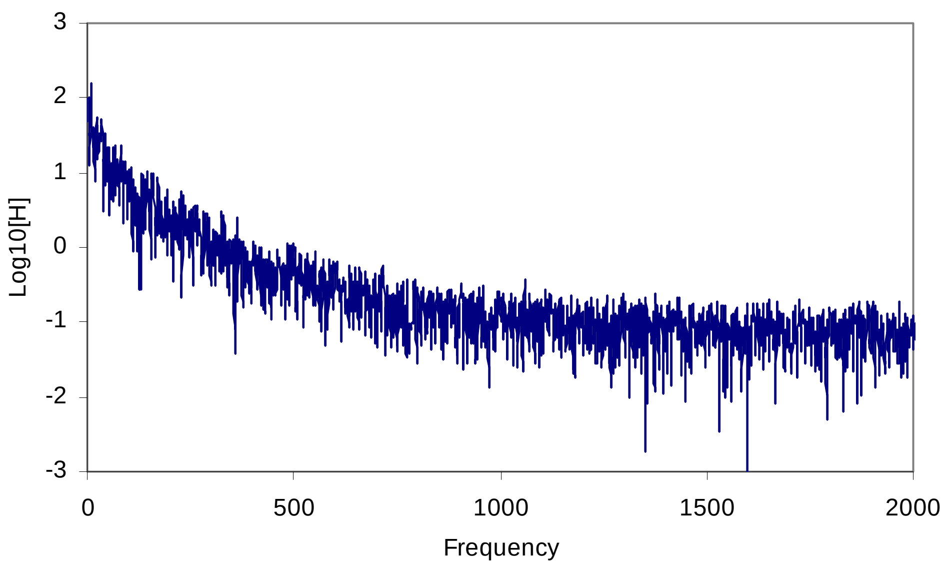
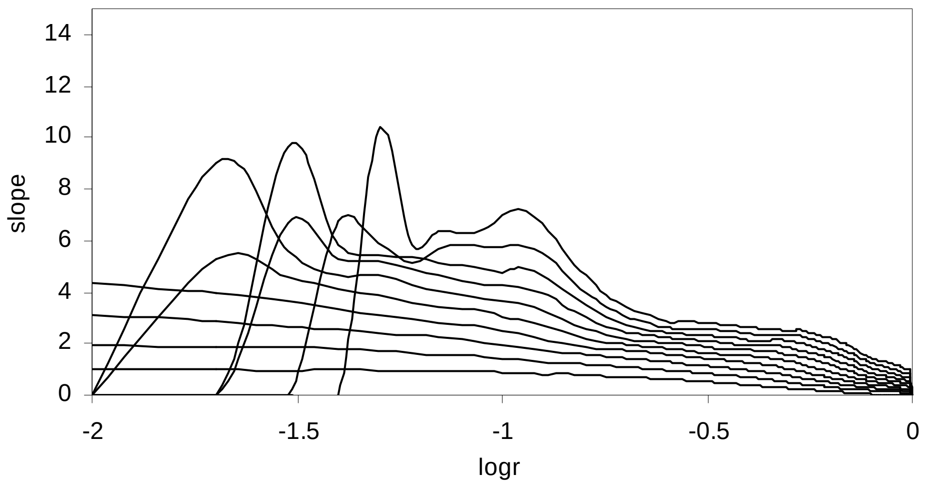


Figure 5.6 Daily time series of Mississippi river flow plotted in different time scales

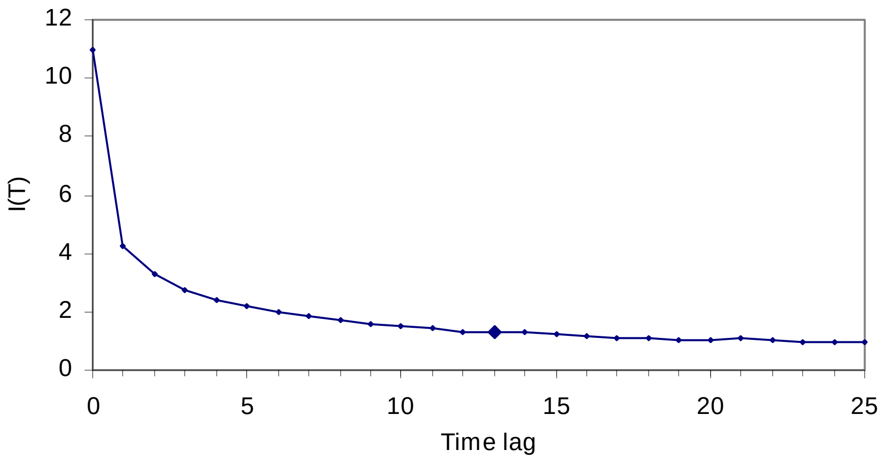


(a) Fourier transform

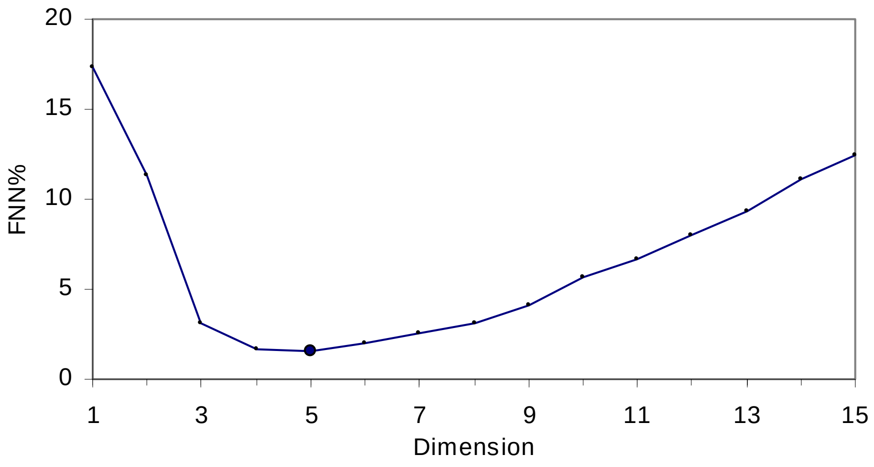


(b) Correlation dimension

Figure 5.7 Fourier transform and correlation dimension of daily Mississippi river flow time series



(a) Average Mutual Information



(b) False Nearest Neighbours

Figure 5.8 Determination of time lag and embedding dimension: Mississippi river time series

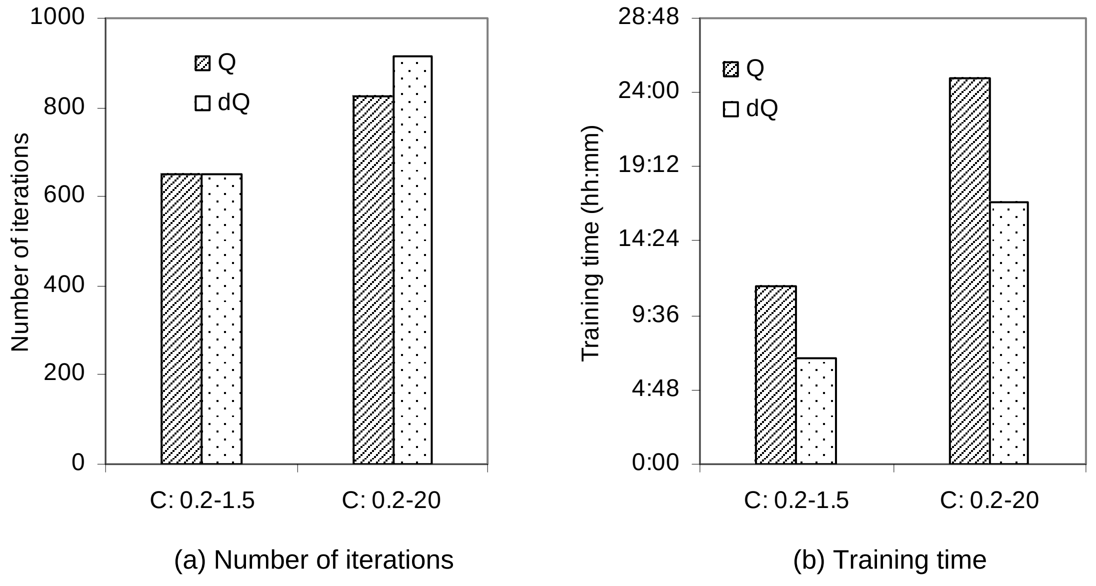


Figure 5.9 Effect of C-range on number of iterations and training time: Tryggevælde catchment runoff time series

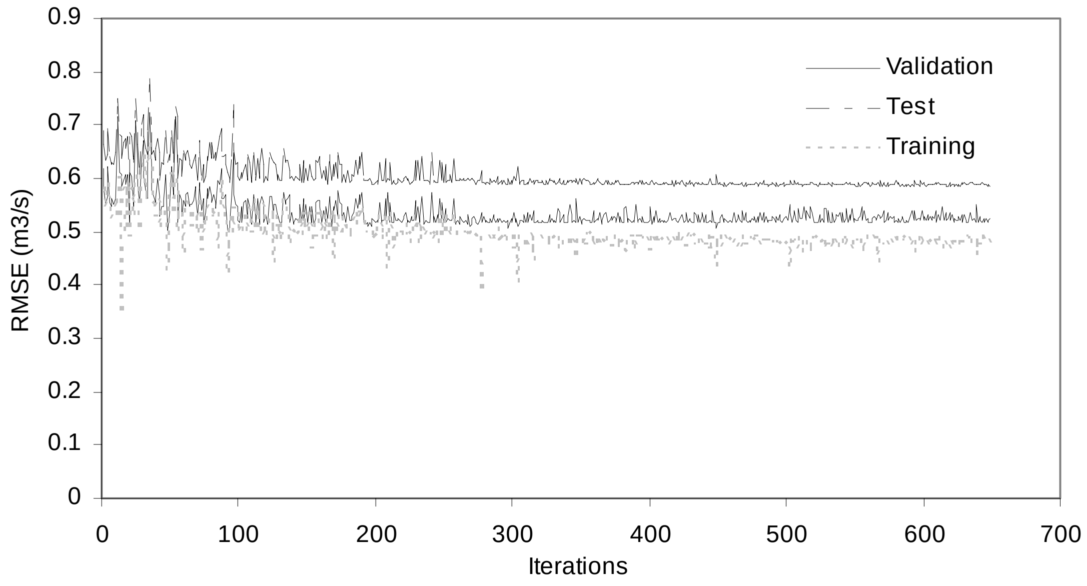


Figure 5.10 Computational convergence of EC-SVM I: Tryggevælde catchment runoff

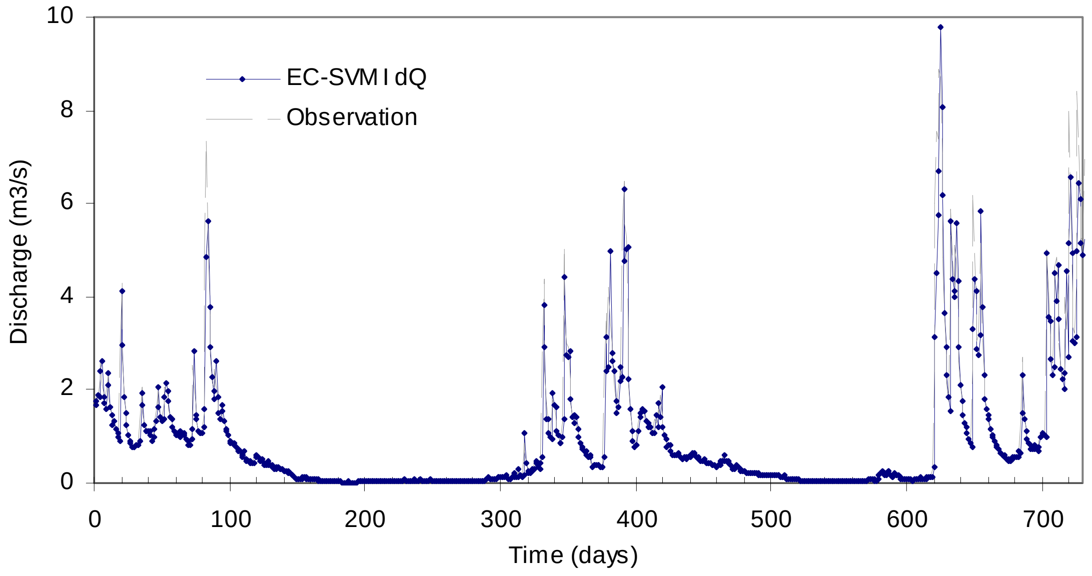


Figure 5.11 Comparison between observed and predicted hydrographs using dQ time series in training: validation set of Tryggevælde catchment runoff

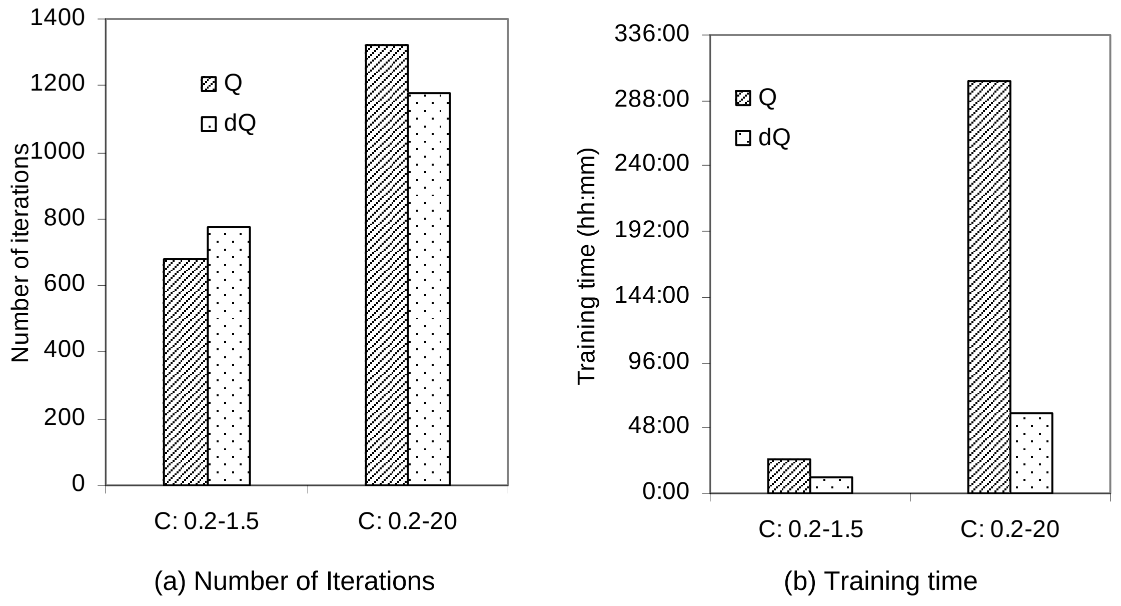


Figure 5.12 Effect of C range on number of iterations and training time of EC-SVM I: Mississippi rive flow

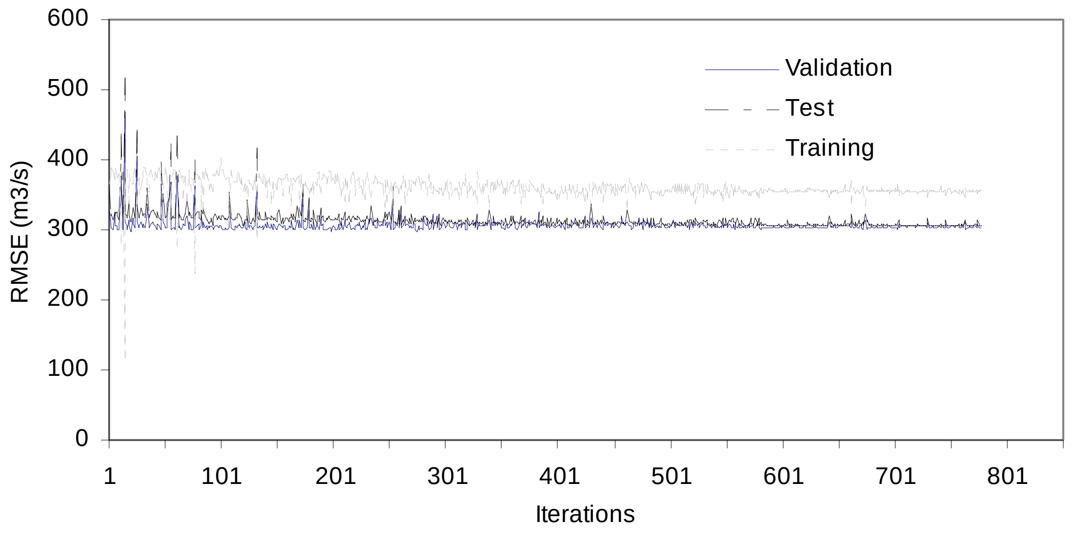


Figure 5.13 Computational convergence of EC-SVM I: Mississippi river flow

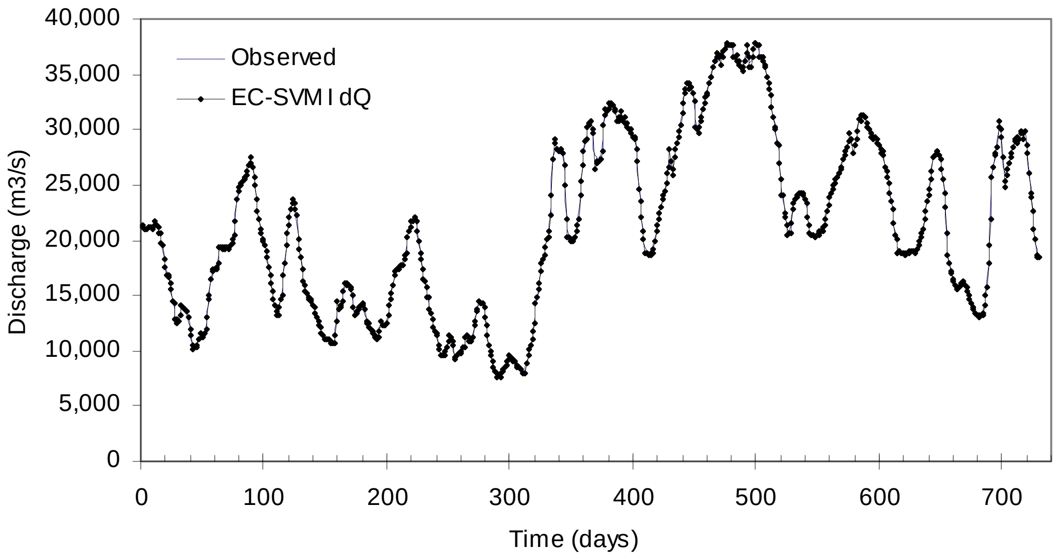


Figure 5.14 Comparison between observed and predicted hydrographs using dQ time series in training: validation set of Mississippi river flow

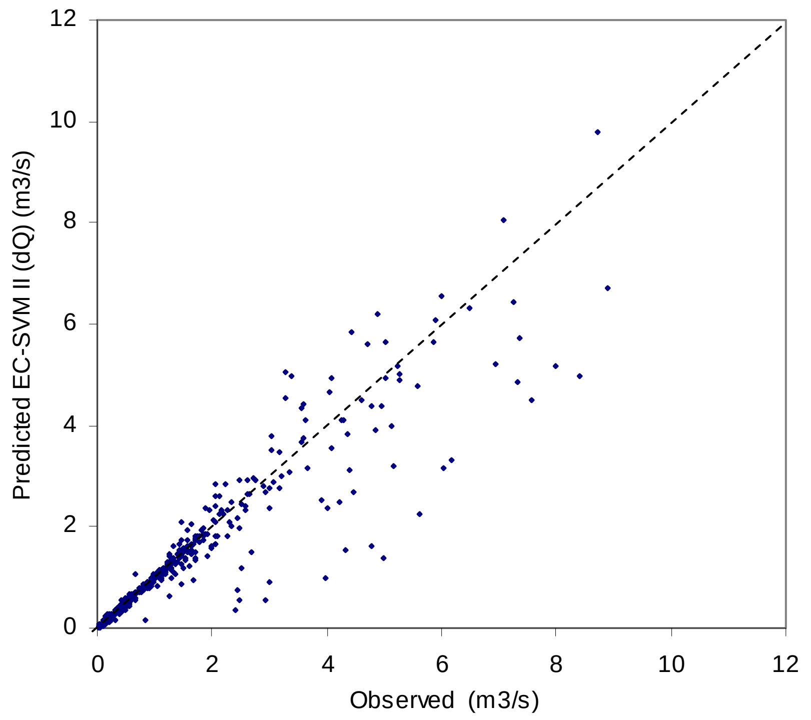


Figure 5.15 Scatter plot of EC-SVM II prediction accuracy using dQ time series: Tryggevælde catchment runoff

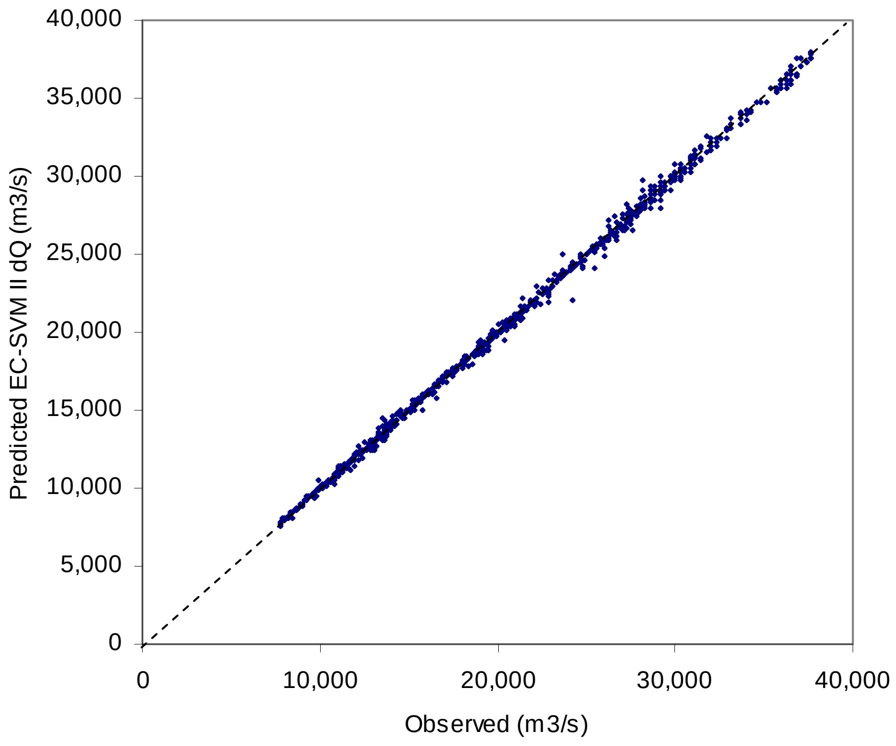
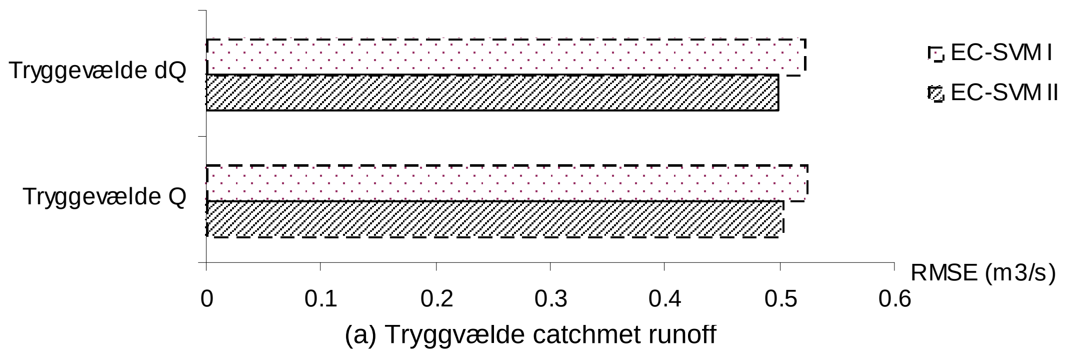


Figure 5.16 Scatter plot of EC-SVM II prediction accuracy using dQ time series: Mississippi river flow



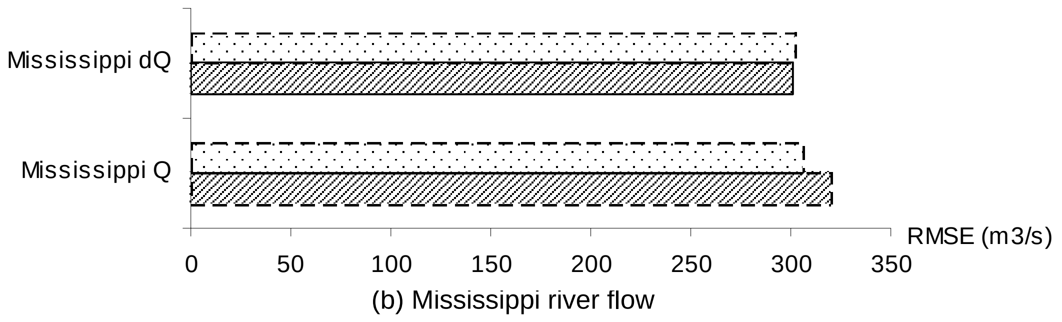
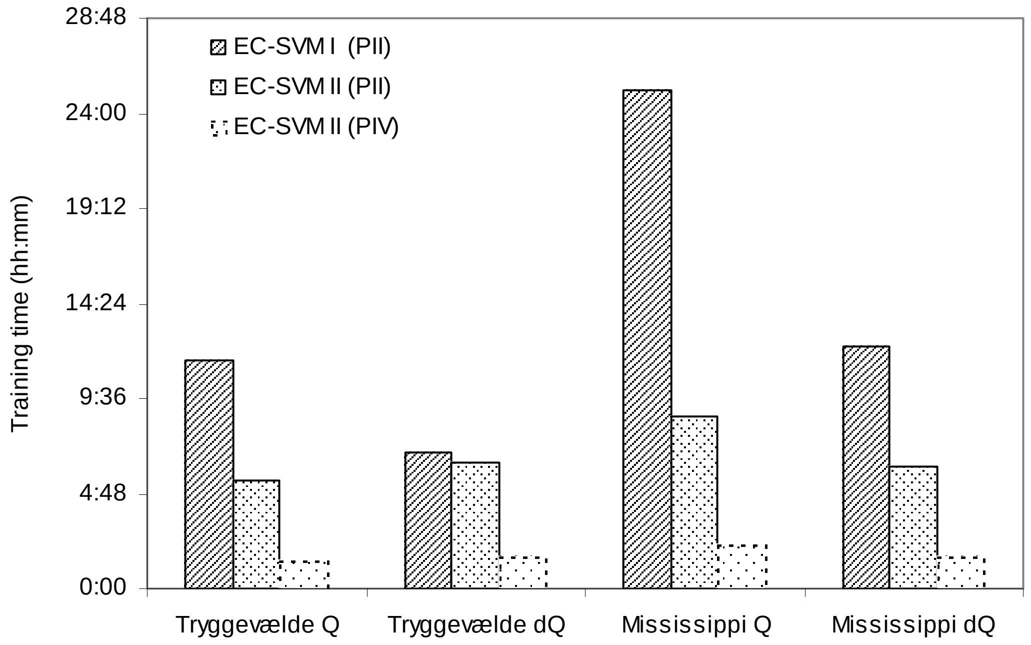
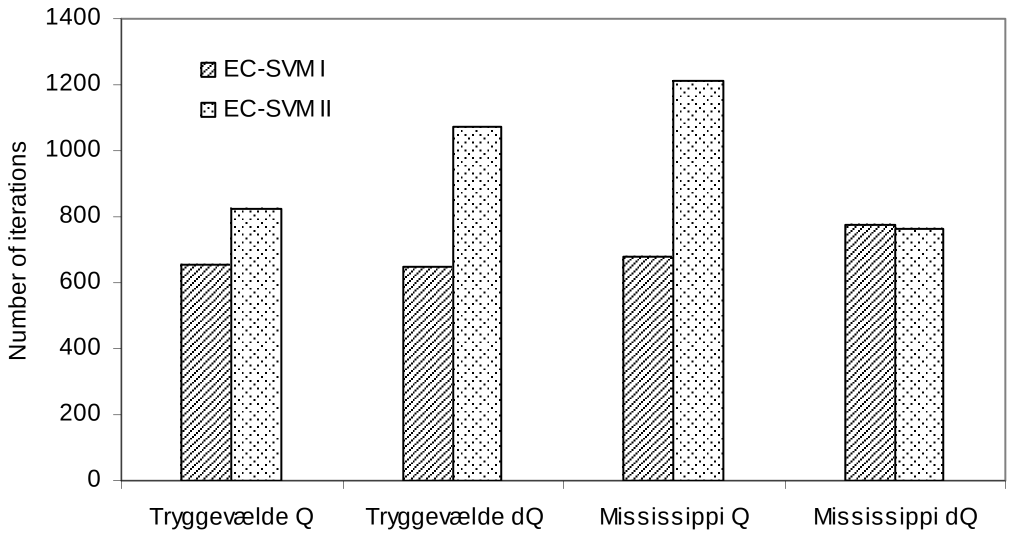


Figure 5.17 Comparison between prediction accuracies resulting from EC-SVM I and EC-SVM II

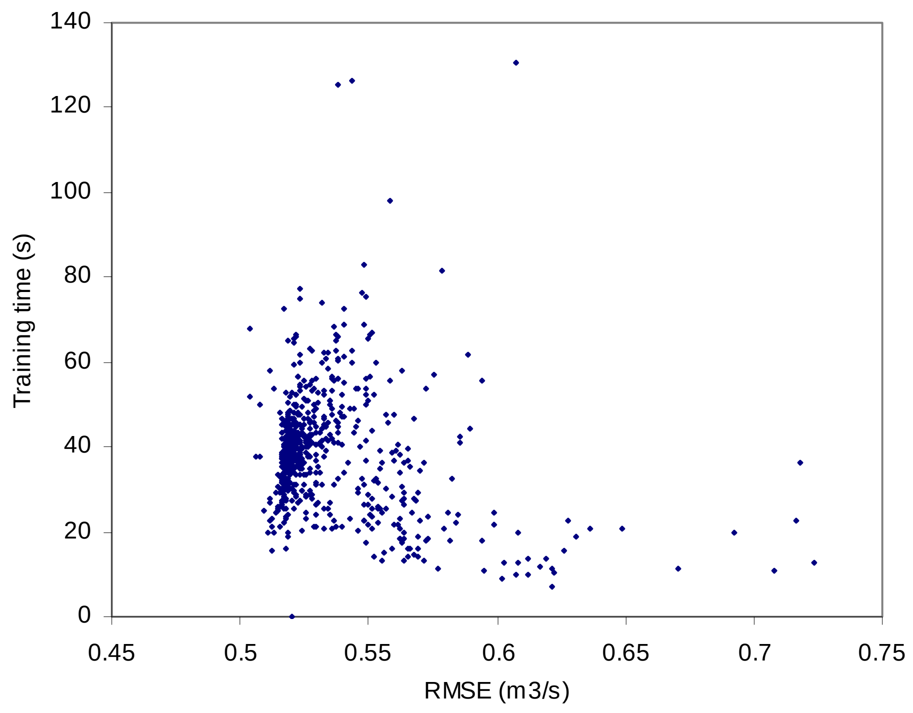


(a) Training time

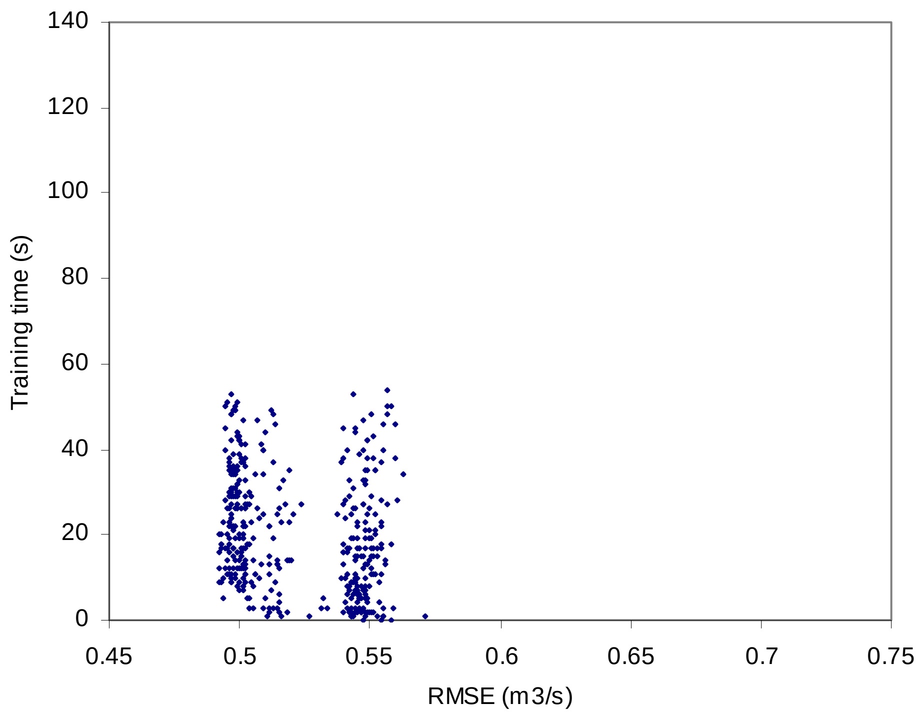


(b) Number of iterations

Figure 5.18 Comparison between computation time and iterations of EC-SVM I and EC-SVM II

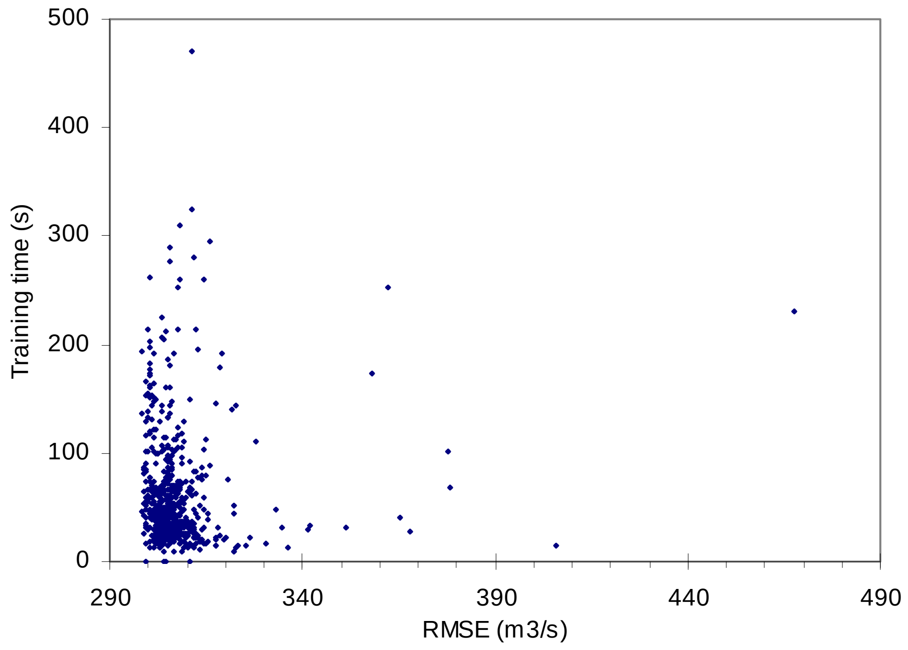


(a) EC-SVM I

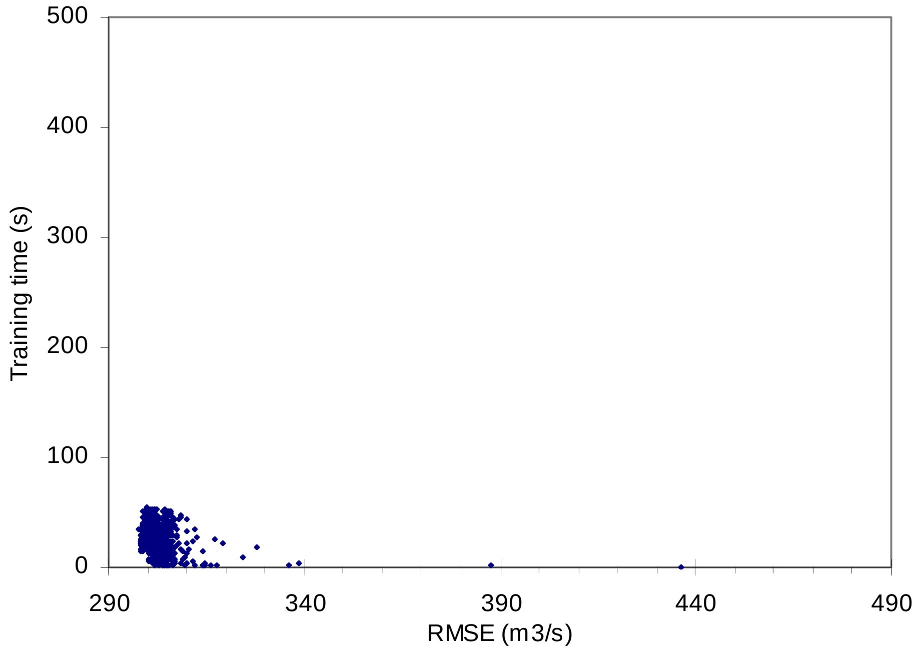


(b) EC-SVM II

Figure 5.19 Prediction accuracy and training time with dQ time series used in training: Tryggevælde catchment runoff



(a) EC-SVM I



(b) EC-SVM II

Figure 5.20 Prediction accuracy and training time with dQ time series used in training: Mississippi river flow

Table 5.1 Range of parameters: EC-SVM I

Parameters	Range 1	Range 2
Delay time τ	1-20	1-20
Embedding dimension d	2-20	2-20
ε-insensitive loss function ε	0-0.1	0-0.1
Gaussian kernel width σ	0.1-0.9	0.1-0.9
Regularisation parameter C	0.2-20.0	0.2-1.5

Table 5.2 Training time and test error of EC-SVM I: Tryggevælde catchment runoff

C-range	Time series used	No. of Iterations	Total time (PII)	RMSE (m^3/s)		
				Training set	Test set	Validation set
[0.2 - 1.5]	Q	652	11h30m	0.500	0.587	0.522
	dQ	649	6h53m	0.481	0.584	0.524
[0.2-20.0]	Q	825	24h55m	0.500	0.587	0.522
	dQ	916	16h55m	0.497	0.595	**0.521**

Table 5.3 Optimal parameter set of EC-SVM I: Tryggevælde catchment runoff

C-range	Time series used	Parameters					Time(s)
		τ	d	C	ε	σ	PII
[0.2 - 1.5]	Q	1	3	1.003	0.003	0.62	104
	dQ	1	6	0.370	0	0.38	62
[0.2-20.0]	Q	1	3	2.170	0.011	0.80	37
	dQ	1	5	5.540	0.016	0.55	48

Table 5.4 Prediction accuracy resulting from various techniques: Tryggevælde catchment runoff

Time series used	Approach	RMSE (m^3/s)	NRMSE	(d,τ)
Q	Standard chaos technique	0.647	0.444	(4,12)
	Naive	0.577	0.396	/
	ARIMA(1,0,1)	0.535	0.367	/
	Inverse approach *	0.527	0.361	(2,1)
	EC-SVM I	**0.522**	**0.358**	**(3,1)**
dQ	Standard chaos technique	0.598	0.41	(4,8)
	ARIMA(1,1,1)	0.543	0.373	/
	EC-SVM I	**0.521**	**0.357**	**(5,1)**

* Liong et al., 2002

Table 5.5 Training time and test error of EC-SVM I: Mississippi river flow

C-range	Time series used	No. of Iterations	Total time (PII)	RMSE (m^3/s)		
				Training set	Test set	Validation set
[0.2 - 1.5]	Q	680	25h12m	389.96	323.31	307.98
	dQ	778	12h12m	355.91	304.74	**302.41**
[0.2-20.0]	Q	1320	302h32m	377.47	310.85	305.81
	dQ	1180	59h03m	358.45	304.45	304.09

Table 5.6 Optimal parameter set of EC-SVM I: Mississippi river flow

C-range	Time series used	Parameters					Time(s)
		τ	d	C	ε	σ	PII
[0.2 - 1.5]	Q	1	5	0.925	0	0.27	169
	dQ	1	3	0.929	0.071	0.29	10
[0.2-20.0]	Q	1	3	13.69	0.001	0.19	1097
	dQ	1	3	4.26	0.034	1.00	94

Table 5.7 Prediction accuracy resulting from various techniques: Mississippi river flow

Time series used	Approach	RMSE (m^3/s)	NRMSE	(d,τ)
Q	Standard chaos technique	1738.95	0.2064	(6,13)
	Naive	608.70	0.0771	/
	ARIMA(1,0,1)	435.00	0.0551	/
	Inverse approach*	356.89	0.0452	(2, 1)
	EC-SVM I	**307.98**	**0.0387**	**(5, 1)**
dQ	Standard chaos technique	365.26	0.0462	(4, 6)
	ARIMA(1,1,1)	322.69	0.0409	/
	EC-SVM I	**302.40**	**0.0385**	**(3, 1)**

* Liong et al., 2002

Table 5.8 Range of the parameters: EC-SVM II

Parameters	*Range*
Delay time τ	*1-20*
Embedding dimension d	*2-20*
Dimension of approximated features q	*5-105*
Gaussian kernel width σ	*0.1-0.9*
Regularisation parameter C'	*0.1-50.0*

Table 5.9 Training time and test error of EC-SVM II: Tryggevælde catchment runoff

Time series used	No. of Iterations	Total time (PII)	Total time (PIV)	RMSE (m^3/s)		
				Training set	Test set	Validation set
Q	824	5h25m	1h24m	0.478	0.573	0.502
dQ	1070	6h23m	1h38m	0.468	0.575	0.500

Table 5.10 Optimal parameter set of EC-SVM II: Tryggevælde catchment runoff

Time series used	Parameters					Time(s) (PII)
	τ	d	q	σ	C'	
Q	1	5	71	0.26	2.8	23
dQ	1	6	97	0.13	1.8	42

Table 5.11 Prediction accuracy resulting from various techniques: Tryggevælde catchment runoff

Time series used	Approach	RMSE (m^3/s)	NRMSE	(d,τ)
Q	Standard chaos technique	0.647	0.444	(4,12)
	Naive	0.577	0.396	/
	ARIMA(1,0,1)	0.535	0.367	/
	Inverse approach *	0.527	0.361	(2,1)
	EC-SVM I	0.522	0.358	(3,1)
	EC-SVM II	**0.501**	**0.344**	**(5,1)**
dQ	Standard chaos technique	0.598	0.41	(4,8)
	ARIMA(1,1,1)	0.543	0.373	/
	EC-SVM I	0.521	0.357	(5,1)
	EC-SVM II	**0.500**	**0.343**	**(6,1)**

* Liong et al., 2002

Table 5.12 Training time and test error of EC-SVM II: Mississippi river flow

Time series used	No. of Iterations	Total time (PII)	Total time (PIV)	RMSE (m^3/s)		
				Training set	Test set	Validation set
Q	1214	8h40m	2h14m	402.68	332.21	320.44
dQ	762	6h08m	1h37m	358.22	306.46	300.71

Table 5.13 Optimal parameter set of EC-SVM II: Mississippi river flow

Time series used	Parameter					Time (s) (PII)
	τ	d	q	σ	C′	
Q	1	4	57	0.55	0.6	15
dQ	1	3	100	0.12	15.4	49

Table 5.14 Prediction accuracy resulting from various techniques: Mississippi river flow

Time series used	Approach	RMSE (m^3/s)	NRMSE	(d,τ)
Q	Standard chaos technique	1738.95	0.2064	(6,13)
	Naive	608.70	0.0771	/
	ARIMA(1,0,1)	435.00	0.0551	/
	Inverse approach*	356.89	0.0452	(2, 1)
	EC-SVM I	307.98	0.0387	(5, 1)
	EC-SVM II	**320.44**	**0.0406**	**(4, 1)**
dQ	Standard chaos technique	365.26	0.0462	(4, 6)
	ARIMA(1,1,1)	322.69	0.0409	/
	EC-SVM I	302.40	0.0383	(3, 1)
	EC-SVM II	**300.71**	**0.0381**	**(3, 1)**

* Liong et al., 2002

Table 5.15 Prediction accuracy of EC-SVM I and EC-SVM II

Catchment	Time series	RMSE (m^3/s)	
		EC-SVM I	EC-SVM II
Tryggevælde	Q	0.522	0.502
	dQ	0.521	**0.500**
Mississippi	Q	305.81	320.44
	dQ	302.40	**300.71**

Table 5.16 Computation time of EC-SVM I and EC-SVM II

Catchment	Time series	EC-SVM I		EC-SVM II		
		Iterations	Time (PII)	Iterations	Time (PII)	Time (PIV)
Tryggevælde	Q	652	11h30m	824	5h25m	1h24m
	dQ	649	6h53m	1070	6h23m	1h38m
Mississippi	Q	680	25h12m	1214	8h40m	2h14m
	dQ	778	12h12m	762	6h08m	1h37m

CHAPTER 6
CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

Forecasting of hydrological time series is one of the basic and important tasks in water resources management. Recently a number of studies have shown that hydrological time series have the characteristics of chaotic time series and possess a low correlation dimension. However, the applications of the chaotic techniques are very often limited to local linear learning machines due to the large amount of data required. This obstacle in analysing hydrological time series is now removed with the schemes proposed in this study, EC-SVM I and EC-SVM II.

Both EC-SVM I and EC-SVM II share many commonalities. Both schemes: (1) use one of latest learning machines, viz. Support Vector Machine (**SVM**); (2) operate SVM on the reconstructed phase space which is the space where **C**haos analysis is commonly conducted; and (3) apply an **E**volutionary algorithm to simultaneously optimize the embedding structure parameters and those of SVM. Both schemes differentiate from each other only in the technique to resolve the large data size often required in chaos analysis. EC-SVM I adopts a decomposition method while EC-SVM II applies the ridge regression method to circumvent the large data sample problem.

Both schemes are demonstrated on the Tryggevaelde catchment runoff time series and the Mississippi river flow daily time series. Considerable improvements were obtained, certainly for the Mississippi case.

6.1.1 SVM applied in phase space reconstruction

The present study is the first study that introduces the application of Support Vector Machine (SVM) in the phase space reconstruction. Previous work on applications of SVM in time series has been limited to dynamics reconstruction. There it is inevitable that a unit time lag has to be assumed. As is known from chaos theory, an order in disorder can be found only at a given embedding structure, i.e. with the correct embedding dimension and the correct time lag.

Thus, this study suggests the applications of SVM in the phase space reconstruction instead of directly using the original time series. The most appropriate embedding structure parameters (embedding dimensions and time lag) are derived based on the least prediction accuracy from an unseen data set. The proposed approach in deriving the values of the embedding structure parameters is recommended instead of using the traditional approach, average mutual information for time lag and then the false nearest neighbours for embedding dimensions. A guaranteed higher prediction accuracy resulting from the proposed approach is obvious since it uses the least prediction accuracy as its objective function.

6.1.2 Handling large data sets effectively

One of the most important abilities of any proposed scheme to analyse chaotic time series is its efficiency in dealing with large data sets. It is a norm that chaotic time series analysis requires large sets of data. Normal SVM can't deal with large data sets. Practitioners opt to select a shorter data set at the expense of prediction accuracy of the trained model.

Two efficient algorithms to handle large data records for SVM are introduced. One of them is the decomposition method while the other one is the linear regression with approximated features method.

The decomposition method decomposes the quadratic optimization programming of the dual problem of SVM into a series of quadratic problems each with only 2 variables; this transformation makes the analysis quite straightforward. The 2 variables are selected through a feasible direction method which is based on the maximum gradient. The decomposition method used in this study is the most efficient algorithm of decomposition methods, known at present.

The linear ridge regression, on the other hand, approximates the feature dimension at which the feature space of the original data is linearly related to the output variable. The approximation of features is based on the eigen function of the kernel of the Mercer's theorem and the eigen decomposition of a square matrix. Good representative points are selected on the basis of the maximum entropy; this implies that the set of points are most scattered. Since the linear ridge regression involves no iterative scheme, as in the decomposition method, the speed is even faster and more reliable.

6.1.3 Evolutionary algorithm for parameters optimization

SVM has several parameters which need to be calibrated. SVM can provide good results only when a good set of values of the calibration parameters is chosen. In addition to the SVM parameters, two more parameters need to be calibrated. They are the embedding structure parameters, i.e. the time lag and the embedding dimension.

An evolutionary algorithm is applied to calibrate these parameters simultaneously and automatically. Evolutionary algorithm is a global optimization approach and is especially suitable for the very difficult task where deterministic sequence of iterative solutions based on the gradient or high order statistics of the cost function can not be generated. Evolutionary algorithm only requires the objective function and therefore it is very suitable for this sort of parameter tuning task. The results from traditional methods help to set suitable parameter search range for evolutionary algorithm. The evolutionary algorithm used in this study is the Shuffled Complex Evolution (SCE) algorithm.

EC-SVM I is with the decomposition method while EC-SVM II is with the linear ridge regression; both are equipped with SCE optimization scheme.

6.1.4 High computational performances

The novel approaches suggested in this study, EC-SVM, show both effectiveness (i.e. high prediction accuracy) and efficiency (i.e. high computational speed).

The EC-SVM approaches are demonstrated on two real daily flow time series: Tryggevælde catchment runoff and Mississippi river flow time series. The results obtained by both EC-SVM I and EC-SVM II prove better than naïve forecasting, ARIMA, and other currently used chaotic techniques. Moreover, the study shows that the first difference runoff time series, dQ, should be seriously considered instead of the original Q time series; analysis with the dQ time series yields higher prediction accuracy.

EC-SVM II (with linear ridge regression) is recommended over EC-SVM I (with decomposition method) particularly with respect to stable and fast computational speed. This is to be expected since the linear ridge regression does not involve any iterative algorithm.

The speed of EC-SVM II is attractively fast. It takes about 1-2 hours on P4 2.4GHz and yet yields very high prediction accuracy.

6.2 Recommendations for future study

Recommendations for future research and practical applications are suggested as follows:

(1) Multivariate analysis

Most of the hydrological systems are complex nonlinear dynamical systems. If time series of other sensitive variables are available, e.g. precipitation (P) and temperature (T), the analysis should include these time series. This extra information may further increase the prediction accuracy of the runoff. In this study EC-SVM approaches are demonstrated only on univariate time series. The approach is applicable to multivariate time series as well. The expression can be written as:

$$\begin{aligned} Q_{t+1} = f(&Q_{t-\tau_Q}, Q_{t-2\tau_Q}, \ldots, Q_{t-(d_Q-1)\tau_Q}, \\ &P_{t-\tau_P}, P_{t-2\tau_P}, \ldots, P_{t-(d_P-1)\tau_P}, \\ &T_{t-\tau_T}, T_{t-2\tau_T}, \ldots, T_{t-(d_T-1)\tau_T}) \end{aligned} \tag{6.1}$$

There are obviously more embedding structure parameters, (τ_Q, d_Q, τ_P, d_P, τ_T, d_T) for the above example. SCE is a very efficient optimization scheme and hence can efficiently deal with 20 genes or more.

(2) Multi-objective optimization

The present study has solely used RMSE as a measure of goodness of fit. Other goodness-of-fit measures should be considered. They are, for example, volume error, peak runoff error, percentage of false nearest neighbours, etc. Evolutionary algorithms for multi-objective optimization are available. Elitist non-dominated sorting genetic algorithm (NSGA II) by Deb (2001) is one of the well developed algorithms for multi-objective optimization problems. Applying NSGA II instead of SCE may fit the calibration task for this multi-objective problems.

(3) Gaussian kernel

Gaussian kernel is one the most powerful kernels and a commonly used kernel. This study applies Gaussian kernel as well. Other powerful kernels for regression such as spline kernel may be more suitable for some time series.

(4) Uncertainty

The current study uses RMSE of the test set as a goodness-of-fit measure. It should be noted that it is much more reasonable to use the test error, as a goodness-of-fit measure, than the training error. Nevertheless, this does not guarantee that the resulting 'optimal' model will yield best prediction accuracy on the validation data set. This is perhaps caused by an overfitted model. It is therefore suggested to create another test set for overfitting test.

REFERENCES

1. Abarbanel, H. D. I., Brown, R. and Kadtke, J. B. Prediction in Chaotic Nonlinear Systems: Methods for Time Series with Broadband Fourier Spectra. Physical Review A, 41(4), pp. 1782-1807. 1990.

2. Abarbanel, H. D. I. Analysis of Observed Chaotic Data. Springer-Verlag, NY. 1996.

3. Abbott, M. B. Introducing Hydroinformatics. Journal of Hydroinformatics, 1(1), pp. 3-19. 1999.

4. Alligood, K., Sauer, T. and Yorke, J.A. CHAOS: An Introduction to Dynamical Systems. Springer-Verlag. 1997.

5. Anctil, F., Michel, C., Perrin, C. and Andréassian, V. A Soil Moisture Index as an Auxiliary ANN input for Stream Flow Forecasting. Journal of Hydrology, 286, pp.155-167. 2004.

6. Babovic, V. and Keijzer, M. Forecasting of River Discharges in the Presence of Chaos and Noise. In Coping with Flood. 1999.

7. Babovic, V. Keijzer, M. and Stefasson, M. Optimal Embedding Using Evolution Algorithms. In Proc. 4th International Conference on Hydroinformatics, Iowa City, USA, July 2000a.

8. Babovic, V., Keijzer, M. and Bundzel, M. From Global to Local Modelling: A Case Study in Error Correction of Deterministic Models. In Proc. 4th International Conference on Hydroinformatics, Iowa City, USA, 2000b.

9. Backer, C. T. H. The Numeral Treatment of Integral Equations. Oxford: Clarendon Press. 1977.

10. Brandstater, A. and Swinney, H. L. Strange Attractor in Weakly Turbulent Couette-Talay flow. Phys. Rev. A 35, pp. 2206. 1986.

11. Boser, B. E., Guyon, I. M. and Vapnik, V. N. A Training Algorithm for Optimal Margin Classifiers. In Proc. 5th Annual ACM Workshop on Computational Learning Theory, ed by Haussler, D., pp. 144-152. Pittsburgh, PA, ACM Press. 1992.

12. Cao, L. Y., Mees, A. and Judd, K. Dynamics from Multivariate Time Series. Physica D, 121, pp.65-88. 1998.

13. Cao, L. Y. Practical Method for Determining the Minimum Embedding Dimension of a Scalar Time Series. Physica D, 110, pp. 43-50. 1997.

14. Casdagli, M. Nonlinear Prediction of Chaotic Time Series. Physica D, 35, pp. 335 - 356. 1989.

15. Casdagli, M. Chaos and Deterministic versus Stochastic Non-linear Modelling. Journal of Royal Statistical Society B, 54(2), pp. 303-328. 1991.

16. Casdagli, M., Eubank, S., Farmer, J. D. and Gibson, J. State Space Reconstruction in the Presence of Noise. Physica D, 51, pp. 52-98. 1991.

17. Cherkassky, V. and Ma, Y. Practical Selection of SVM Parameters and Noise Estimation for SVM Regression. Neural Networks, 17(1), pp 113-126. 2004.

18. Cherkassky, V. and Mulier, F. Learning from Data: Concepts, Theory and Methods. John Wiley and Sons. 1998.

19. Collobert, R. and Bengio, S. On the Convergence of SVMTorch, an Algorithm for Large-Scale Regression Problems. Technical Report IDIAP-RR 00-24, IDIAP, Martigny, Switzerland. 2000.

20. Collobert, R. and Bengio, S. SVMtorch: Support Vector Machines for Large-Scale Regression Problems. Journal of Machine Learning Research, 1, pp 143-160. 2001.

21. Collobert, R., Bengio, S., and Bengio, Y., A Parallel Mixture of SVMs for Very Large Scale Problems. Neural Computation, 14(5) pp.1105-1114. 2002.

22. Cover, T. and Hart, P. Nearest Neighbour Pattern Classification. IEEE Transactions on Information Theory, 13, pp.21-27. 1967.

23. Doan, C. D., Liong, S. Y. and Karunasingha, D. S. K. Deriving Effective and Efficient Data Set with Subtractive Clustering Method and Genetic Algorithm. Submitted to Journal of Hydroinformatics. 2003.

24. Deb, K. Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley &Sons. 2001.

25. Dibike Y. B., Velickov S., Solomatine D. P. and Abbott M. B. Model Induction with Support Vector Machines: Introduction and Applications. Journal of Computing in Civil Engineering, American Society of Civil Engineers (ASCE), 15(3), pp. 208-216. 2001.

26. Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J. and Vapnik, V. Support vector regression machines. In: Advances in Neural Information Processing Systems, MIT Press, Cambridge M.A., pp.155-161. 1997.

27. Duan, Q., Sorooshian, S. and Gupta, V. K. Effective and Efficient Global Optimization for Conceptual Rainfall-Runoff Models. Water Resour. Res. 28(4), pp.1015-1031. 1992.

28. Duda, R. O. and Hart, P. E. Pattern Classification and Scene Analysis. Wiley, New York. 1973.

29. Eckmann, J. P., Kamphorst, S. O., Ruelle, D. and Ciliberto, S. Lyapunov Exponents from Time Series. Physical Review A, 34(6), pp. 4971-4979. 1986.

30. Essex, C., Lookman, T. and Nerenberg, M. A. H. The Climate Attractor over Short Timescales. Nature, 326, pp. 64-66. 1987.

31. Espinoza, M., Suykens, J. and De Moor, B. Least Squares Support Vector Machines and Primal Space Estimation. In Proc. IEEE 42nd Conference on Decision and Control, Maui, USA, 2003 December.

32. Fan, J. D. and Sidorowich, J. J. Local Polynomial Modelling and its Applications. Chapman & Hall, London, UK. 1996.

33. Farmer, J. D. and Sidorowich, J. J. Predicting Chaotic Time Series. Phys. Rev. Lett. 59, pp. 845-848. 1987.

34. Fraedrich, K. Estimating the Dimensions of Weather and Climate Attractors. Journal of the Atmospheric Sciences, 43(5), pp. 419-432. 1986.

35. Fraedrich, K. Estimating Weather and Climate Predictability on Attractors. Journal of the Atmospheric Sciences, 44 (4), pp. 722-728. 1987.

36. Frazer, A. M. Reconstructing Attractors from Scalar Time series: A Comparison of Singular System and Redundancy Criteria. Physica D, 34, pp.391-404. 1989.

37. Fogel, D. B. An Introduction to Simulated Evolutionary Optimization. IEEE Trans. Neural Networks, 5(1), pp. 3-14. 1994.

38. Fogel, L. J., Owens, A. J. and Walsh, M. J. Artificial Intelligence through Simulated Evolution, New York: John Wiley. 1966.

39. Fraser, A. and Swinney, H. Independent Coordinates for Strange Attractors from Mutual Information. Phys. Rev. A 33, pp. 1134-1140. 1986.

40. Frison, T. Nonlinear Data Analysis Techniques. In: Trading on the Edge. Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets, ed by Deboeck, G. J., pp. 280-296. John Wiley Inc., New York. 1994.

41. Geman, S., Bienenstock, E. and Doursat, R. Neural Networks and the Bias/Variance Dilemma. Neural Computation 4, pp. 1-58. 1992.

42. Gershenfeld N. and Weigend, A. The Future of Time Series: Learning and Understanding. In Time Series Prediction: Forecasting the Future and Understanding the Past, ed by Weigend, A. and Gershenfeld, N., pp.1-70. Addison Wesley. 1993.

43. Gibson, J. F., Farmer, J. D., Casdagli, M. and Eubank, S. An Analytical Approach to Practical State Space Reconstruction. Physica D, 57, pp. 1-30. 1992.

44. Girolami, M. Orthogonal Series Density Estimation and the Kernel Eigenvalue Problem. Neural Computation, 14, pp. 669-688. 2002.

45. Gleick, J. Chaos: Making a New Science. Viking Penguin, New York. 1987.

46. Grassberger, P. and Procaccia, I. Characterization of Strange Attractors, Phys. Rev. Lett., 50, pp. 346. 1983a.

47. Grassberger, P. and Procaccia, I. Measuring the Strangeness of Strange Attractors. Physica D, 9, pp.189-208. 1983b.

48. Grassberger, P. and Procaccia, I. Estimation of the Kolmogorov Entropy from a Chaotic Signal. Physical Review A, 28, pp. 2591-2593. 1983c.

49. Grassberger, P. Do Climatic Attractors Exist? Nature, 323, pp. 609-612. 1986.

50. Haykin, S. Neural Networks: A Comprehensive Foundation, 2nd edition. Prentice-Hall, New Jersey. 1999.

51. Hense, A. On the Possible Existence of a Strange Attractor for the Southern Oscillation. Beitr. Phys. Atmosphere, 60(1), pp. 34-47. 1987.

52. Hilborn, R.C. Chaos and Nonlinear Dynamics, pp. 40. Oxford University Press. 1994.

53. Holland, J. H. Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press. 1975.

54. Holzfuss, J. and Mayer-Kress, G. An Approach to Error-estimation in the Application of Dimension Algorithms. In Dimensions and Entropies in Chaotic Systems, ed by Mayer-Kress, G., pp. 114-122. Springer-Verlag, New York. 1986.

55. Ikeguchi, T. and Aihara, K. Prediction of Chaotic Time Series with Noise. IEEE Transactions, Fundamentals, E78 (10), pp. 1291-1297. 1995.

56. Islam, S., Bras, R. L. and Rodriguez-Iturbe, I. A Possible Explanation for Low Correlation Dimension Estimates for the Atmosphere. Journal of Applied Meteorology, 32, pp. 203-208. 1993.

57. Izenman, A. J. Recent Developments in Nonparametric Density Estimation. Journal of the American Statistical Association, 86, pp. 205-224. 1991.

58. Jayawardena, A. W. and Lai, F. Analysis and Prediction of Chaos in Rainfall and Stream Flow Time Series. Journal of Hydrology, 153, pp. 23-52. 1994.

59. Jayawardena, A. W. and Gurung A. B. Noise Reduction and Prediction of Hydrometeorological Time Series: Dynamical Systems Approach vs. Stochastic Approach. Journal of Hydrology, 228, pp. 242-264. 2000.

60. Joachims, T. Making Large-Scale SVM Learning Practical. In: Advances in Kernel Methods - Support Vector Learning, ed by Schölkopf, B., Burges, C. and Smola A., pp. 169-183. MIT Press. 1999.

61. Karunanithi, N., Grenney, W.J., Whitley, D. and Bovee, K. Neural Networks for River Flow Prediction. J. Comput. Civil Engng., 8(22), pp. 201-220. 1994.

62. Kennel, M.B., Brown, R., and Abarbanel, H. D. I. Determining Embedding Dimension for Phase-Space Reconstruction Using a Geometrical Construction. Phys. Rev. A 45, pp. 3403-3411. 1992.

63. Keerthi, S. S., Shevade, S. K., Bhattacharyya C. and Murthy, K. R. K. Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation, 13, pp. 637-649. 2001.

64. Keerthi, S. S. and Gilbert, E. G. Convergence of a Generalized SMO Algorithm for SVM Classifier Design. Machine Learning, 46, pp. 351-360. 2002.

65. Krishnakumar, K. Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization. SPIE: Intelligent Control and Adaptive Systems, 1196. Philadelphia, PA. 1989.

66. Kuczera, G. Efficient Subspace Probabilistic Parameter Optimization for Catchment models. Water Resour. Res. 33(1), pp.177-185. 1997.

67. Kugiumtzis, D., Lillekendlie, B. and Christophersen, N. Chaotic Time Series Part I: Estimation of Some Invariant Properties in State Space. Modeling, Identification & Control 15(4), pp. 205-224. 1995.

68. Kwok, J. T. Linear Dependency between ε and the Input Noise in ε-Support Vector Regression. In: Proc. International Conference Artificial Neural Networks - ICANN 2001, ed by G. Dorffner, H. Bischof, K. Hornik, pp. 405-410. Lecture Notes in Computer Science 2130 Springer 2001, ISBN 3-540-42486-5.

69. Laskov, P. An Improved Decomposition Algorithm for Regression Support Vector Machines. In Advances in Neural Information Processing Systems 12, ed by Solla, S.A., Leen, T.K. and Müller, K.-R., pp. 484-490. MIT Press. 2000.

70. Laskov, P. Feasible Direction Decomposition Algorithms for Training Support Vector Machines. Machine Learning, Special Issue on Support Vector Machines. 2001.

71. Liebert, W., Pawelzik, K. and Schuster, H. G. Optimal Embeddings of Chaotic Attractors from Topological Considerations. Europhys. Lett., 14, pp. 521-526. 1991.

72. Liong, S. Y., Chan, W. T. and Shreeram, J. Peak Flow Forecasting with Genetic Algorithm and SWMM. Journal of Hydraulic Engineering, ASCE, 121(8), pp. 613-617. 1995.

73. Liong, S. Y., Khu, S. T. and Chan, W. T. Derivation of Pareto Front with Genetic Algorithm and Neural Network? Journal of Hydrologic Engineering, ASCE, 6(1), pp. 56-61. 2001.

74. Liong, S. Y., Lim, W. H., and Paudyal, G. Real Time River Stage Forecasting for Flood Stricken Bangladesh: Neural Network Approach. Journal of Computing in Civil Engineering, ASCE, 4(1), pp. 38-48. 1999.

75. Liong, S. Y. and Sivapragasam, C. Flood Stage Forecasting with SVM. J. Am. Water Res. Assoc., 38(1), pp. 173-186. 2002.

76. Liong, S. Y., Phoon, K. K., Pasha, M. F. K and Doan, C. D. A Robust and Efficient Scheme in Search for Optimal Prediction Parameters Set in Chaotic Time Series. First Asia Pacific DHI Software Conference, Bangkok, (keynote paper). 2002.

77. Lorenz, E. N. Deterministic Nonperiodic Flow. J. Atmos. Sci., 20, pp.130-141. 1963.

78. MacKay, D. J. C. Introduction to Gaussian Processes. Extended Version of a Tutorial at ICANN'97, ftp://wol.ra.phy.cam.ac.uk/pub/mackay/gpB.ps.gz , 1997.

79. Maidment, D. R. (ed). Handbook of Hydrology. U.S.A: McGraw-Hill, Inc. 1993.

80. Matterra, D. and Haykin, S. Support Vector Machines for Dynamic Reconstruction of a Chaotic System. In: Advances in Kernel Methods, ed by Chölkopf, B., Burges, C. J. C and Smola, A. J., pp. 211-241. MIT Press. 1999.

81. Mees, A. I., Rapp, P. E. and Jennings, L. S. Singular Value Decomposition and Embedding Dimension. Phys. Rev. A 36, pp. 340-346. 1987.

82. Muller, K. R., Smola, A., Ratsch, G., Scholkopf, B., Kohlmorgen, J. and Vapnik, V. Predicting Time Series with Support Vector Machines. In Proc. International Conferenceon Artificial Neural Networks, pp.999. Springer Lecture Notes in Computer Science. 1997.

83. Nelder, J. A. and Mead, R. A Simplex Method for Function Minimization. Comput. J. 7, pp.308-313. 1965.

84. Neal, R. M. Regression and Classification Using Gaussian Process Priors (with discussion). In Bayesian Statistics 6, ed by Bernardo, J. M., Berger, J. O., Dawid, A. P. and Smith, A. F. M., pp. 475-501. Oxford University Press. 1999.

85. Nicolis, C. and Nicolis, G. Is There a Climatic Attractor? Nature, 311, pp. 529-532. 1984.

86. Ogawa, H. and Oja, E. Can We Solve the Continuous Karhunen-Loeve Eigenproblem from Discrete Data? Trans. IECE Japan E69, pp. 1020-1029. 1986.

87. Omohundro, S. M., Efficient Algorithms with Neural Network Behaviour. Complex system 1, pp. 273-347. 1987.

88. Osborne, A. R. and Provenzale, A. Finite Correlation Dimension for Stochastic Systems with Power-law Spectra, Physica D, 35, pp. 357-381. 1989.

89. Osuna, E., Freund, R. and Girosi, F. An Improved Training Algorithm for Support Vector Machines. In Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop, ed by Principe, J., Gile, L., Morgan, N. and Wilson, E., pp. 276-285. New York. 1997a.

90. Osuna, E., Freund, R. and Girosi, F. Training Support Vector Machines: An Application to Face Detection. In Proc. Computer Vision and Pattern Recognition '97, pp. 130-136. 1997b.

91. Ott, E., Sauer, T. and Yorke, J. Coping with Chaos. John Wiley & Sons, NY. 1994.

92. Packard, N. H., Crutchfield, J. P., Farmer, J. D. and Shaw, R. S. Geometry from a Time Series. Physical Review Letters, 45(9), pp. 712-716. 1980.

93. Phoon, K. K., Islam, M. N., Liaw, C. Y. and Liong, S. Y. A Practical Inverse Approach for Forecasting Nonlinear Hydrological Time Series. Journal of Hydrologic Engineering, ASCE, 7 (2), pp. 116-128. 2002.

94. Platt, J. C. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In: Advances in Kernel Methods - Support Vector Learning, ed by Schölkopf, B., Burges, C. and Smola, A., pp. 185-208. MIT Press, 1999.

95. Porporato, A. and Ridolfi, L. Clues to the Existence of Deterministic Chaos in River Flow. Journal of Modern Physics B, 10(5), pp. 1821-1862. 1996.

96. Porporato, A. and Ridolfi, L. Nonlinear Analysis of River Flow Time Sequences. Water Resources Research, 33(6), pp. 1353-1367. 1997.

97. Prichard, D. and Theiler, J. Generalised Redundancies for Time Series Analysis. Physica D, 84, pp.476-493. 1995.

98. Rechenberg, I. Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart, Frommann-Holzboog. 1973.

99. Rodriguez-Iturbe, I., De Power, B. F., Sharifi, M. B. and Georgakakos, K. P. Chaos in Rainfall. Water Resources Research, 25(7), pp. 1667-1775. 1989.

100. Sangoyomi, T. B., Lall, U. and Abarbanel, H. D. I. Nonlinear Dynamics of the Great Salt Lake: Dimension Estimation. Water Resources Research, 32(1), pp. 149-159. 1996.

101. Samet, H. The Quadtree and Related Hierarchical Data structures. Computing Surveys, 16(2). 1984.

102. Sauer, T. Yorke, J. and Casdagli, M. Embedology. Journal of Statistical Physics, 65(3/4), pp. 579-616. 1991.

103. Sauer, T. A Noise Reduction Method for Signals from Nonlinear Systems. Physica D, 58, pp. 193- 201. 1992.

104. Schölkopf, B., Smola, A. and Muller, K. R. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Comp. 10, pp. 1299-1319. 1998a.

105. Schölkopf, B., Bartlett, P., Smola, A. and Williamson, R. Support Vector Regression with Automatic Accuracy Control. In Proceedings of ICANN'98, Perspectives in Neural Computing, Berlin, ed by Niklasson, L., Bodén, M. and Ziemke, T., pp.111-116. Springer Verlag. 1998b.

106. Schölkopf, B., Simard, P. Y., Smola, A. J. and Vapnik, V. N. Prior Knowledge in Support Vector Kernels. In Advances in Neural Information Processing Systems, Vol. 10, ed by Jordan, M. I., Kearns, M. J. and Solla, S. A., pp. 640-646. MIT Press, Cambridge, MA. 1998c.

107. Schölkopf, B., Smola, A. and Müller, K.-R. Kernel Principal Component Analysis. In Advances in Kernel Methods - SV Learning, ed by Schölkopf, B., Burges, C. J. C. and Smola, A. J. , pp. 327-352. MIT Press. 1999a.

108. Schölkopf, B., Mika, S., Burges, C.J.C., Knirsch, P., Müller, K.-R., Rätsch, G. and Smola, A. Input Space vs. Feature Space in Kernel-based Methods. IEEE Transactions on Neural Networks, 10(5), pp.1000-1017. 1999b.

109. Schölkopf, B., Smola, A., Williamson, R. and Bartlett, P. L. New Support Vector Algorithm. Neural Computation, 12(5), pp.1207-1245. 2000.

110. Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J. and Williamson, R. C. Estimating the Support of a High-dimensional Distribution. Neural Computation, 13(7), pp. 1443-1472. 2001.

111. Schölkopf, B. and Smola, A. Learning with Kernels. MIT Press. 2002.

112. Schuster, H. G. Deterministic Chaos. VCH Weinheim, Germany. 1988.

113. Schwefel, H.-P. Numerical Optimization of Computer Models. Chichester: Wiley & Sons. 1981.

114. Sharifi, M. B., Georgakakos, K. P. and Rodriguez-Iturbe, I. Evidence of Deterministic Chaos in the Pulse of Storm Rainfall. Journal of the Atmospheric Sciences, 47(7), pp.888-893. 1990.

115. Shevade, S. K., Keerthi, S. S., Bhattacharyya, C. and Murthy, K. R. K. Improvements to the SMO Algorithm for SVM Regression. IEEE Transactions on Neural Networks, 11, pp.1188-1194. 2000.

116. Sivakumar, B., Phoon, K.K., Liong, S.Y., and Liaw, C.Y., A Systematic Approach to Noise Reduction in Chaotic Hydrological Time Series. Journal of Hydrology, 219, pp.103-135. 1999.

117. Sivakumar, B., Liong S.Y., and Liaw, C.Y. Evidence of Chaotic Behaviour in Singapore Rainfall, Journal of American Water Resources Association, 34(2), pp. 301-310. 1998.

118. Sivapragasam, C. Multi-Objective Evolutionary Techniques in Defining Optimal Policies for Real Time Operation of Reservoir Systems. PhD thesis, National University of Singapore. 2003.

119. Smola, A. J., Murata, N., Schölkopf, B. and Müller, K. Asymptotically Optimal Choice of ε-loss for Support Vector Machines. In: Proc. 8th International Conference on Artificial Neural Networks, pp. 105-110. Springer-Verlag. 1998a.

120. Smola, A. J. Learning with Kernels. PhD thesis, Technische Universität Berlin. 1998b.

121. Smola, A. J. and Schölkopf, B. From Regularization Operators to Support Vector Kernels. In Advances in Neural information processings systems 10, San Mateo, CA, pp. 343-349. 1998c.

122. Smola, A. J., Frieß, T. and Schölkopf, B. Semiparametric Support Vector and Linear Programming Machines. In Advances in Neural Information Processing Systems, 11. MIT Press. 1998d.

123. Smola, A. J., Schölkopf, B. and Müller, K.-R. The Connection between Regularization Operators and Support Vector Kernels. Neural Networks, 11, pp.637-649. 1998e.

124. Sugihara, G. and May, R.M. Nonlinear Forecasting as a Way of Distinguishing Chaos from Measurement Error in Time Series, Nature, 344, pp.734-741. 1990.

125. Suykens J. A. K., Lukas L., Van Dooren P., De Moor B. and Vandewalle J. Least Squares Support Vector Machine Classifiers: a Large Scale Algorithm. In Proc. of the European Conference on Circuit Theory and Design (ECCTD'99), Stresa, Italy, Sep. 1999, pp. 839-842.

126. Suykens, J. A. K., Gestel, T. Van, Brabanter, J. De, Moor, B. De and Vandewalle, J. Least Squares Support Vector Machines. World Scientific Pub. Co., Singapore. 2002.

127. Takens, F. In: Dynamical Systems and Turbulence, Vol. 898 of Lecture Notes in Mathematics (Warwick), ed by Rand A. and Young L.S., p366. Springer. 1981.

128. Termonia, Y. and Alexandrovicz, Z. Fractal Dimension of Strange Attractors from Radius versus Size of Arbitrary Clusters. Physical Review Letters, 51, pp. 1265-1268. 1983.

129. Theiler, J. Efficient Algorithm for Estimating the Correlation Dimension from a Set of Discrete Points. Physical Review A, 36(9), pp. 4456- 4462. 1987.

130. Tsonis, A. A. and Elsner, J. B. The Weather Attractor over Very Short Time Scales. Nature, 333, pp. 545-547. 1988.

131. Tsonis, A. A. and Elsner, J. B. Nonlinear Prediction as a way of Distinguishing Chaos from Random Fractal Sequences. Nature, 358, pp. 217-220. 1992.

132. Toth, E., Brath, A. and Montanari, A. Comparison of Short-term Rainfall Prediction Models for Real-time Flood Forecasting. Journal of Hydrology, 239, pp. 132-147. 2000.

133. Vapnik, V. N. Principle of Risk Minimization for Learning Theory. Advances in Neural Information Processing System 4, San Meteo, CA, pp. 831-838. 1992.

134. Vapnik, V., Golowich, S. and Smola, A. Support Vector Method for Function Approximation, Regression Estimation, and Signal processing. In Advances in Neural Information Processing Systems 9, Cambridge, MA, ed by Mozer, M., Jordan, M. and Petsche, T., pp.281-287. MIT Press. 1997.

135. Vapnik, V. Statistical Learning Theory. Wiley, NY. 1998.

136. Weigend, A. S. and Gershenfeld, N. A. The Future of Time Series: Learning and Understanding. In Time Series Prediction: Forecasting the Future and Understanding the Past: Proc. NATO Advanced Research Workshop on Comparative Time Series Analysis, 1994. ed by Weigend A. S. and Gershenfeld, N. A..

137. Williams, C. K. I. Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond. In Learning in Graphical Models, ed by Jordan, M. I., pp. 599-621. Kluwer Academic. 1998.

138. Williams, C. K. I. and Barber, D. Bayesian Classification with Gaussian Processes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12), pp.1342-1351. 1998.

139. Williams, C. K. I. and Seeger, M. The Effect of the Input Density Distribution on Kernel-based Classifiers. In Proceedings of the Seventeenth International Conference on Machine Learning. 2000.

140. Williams, C., K., I., and Seeger, M. Using the Nystrom Methods to Speed Up Kernel Machines. In: Advances in Neural information Processing Systems, 13, pp.682-688. MIT Press. 2001.

141. Wolf, A., Swift, J. B., Swinney, H. L. and Vastano, A. Determining Lyapunov Exponents from a Time Series. Physica D, 16, pp. 285-317. 1985.

142. Zealand, C. M., Burn D. H. and Simonovic, S. P. Short Term Streamflow Forecasting Using Artificial Neural Networks, Journal of Hydrology, 214, pp. 32-48. 1999.

143. Zaldivart, J. M., Gutierrez, E., Galvan, I. M., Strozzi, F. and Tomasin, A. Forecasting High Water Level at Venice Lagoon Using Chaotic Time Series Analysis and Nonlinear Neural Network. Journal of Hydroinformatics, 2, pp. 61-84. 2000.

144. Zhu, H., Williams, C. K. I., Rohwer, R. J. and Morciniec. M. Gaussian Regression and Optimal Finite Dimensional Linear Models. In Neural Networks and Machine Learning, ed by Bishop, C. M., Springer-Verlag, Berlin. 1998.

145. Zoutendijk, G. Methods of Feasible Directions: a Study in Linear and Non-linear Programming. Elsevier. 1970.

146. EPA (U.S. Environmental Protection Agency), http://www.epa.gov/.

147. Krak, http://www.krak.dk/

148. Kernel machine web page, http://www.kernel-machines.org/.

149. LAPACK, http://www.netlib.org/lapack/.

150. LS-SVMlab, http://www.esat.kuleuven.ac.be/sista/lssvmlab/.

151. SVM Torch II, http://www.idiap.ch/learning/SVMTorch.html.

152. USGS (U.S. Geological Survey), http://www.usgs.gov/

LIST OF PUBLICATIONS

Part of this thesis have been published in or submitted for possible publication to the following international Journals or conferences:

Keynote Paper

Liong, S. Y. and **Yu, X. Y.** Support Vector Machine in Chaotic Time Series Forecasting. 28-th International Hydrology and Water Resources Symposium, Australia, 10 – 13 November 2003.

International Journals

- **Yu, X. Y.**, Liong, S. Y., and Babovic, V. EC-SVM Approach For Real Time Hydrologic Forecasting. Journal of Hydroinformatics, V6 (3), pp 209-223. 2004.
- **Yu, X. Y.** and Liong, S. Y. Forecasting of Hydrologic Time Series with Ridge Regression in Feature space of Gaussian Kernel. Submitted for possible publication in Journal of Hydrology. 2004.
- Liong, S. Y., MD. Atiquzzaman and **Yu, X. Y.** Alternative Decision Making in Water Distribution Network with NSGA-II. Submitted for Possible Publication in Journal of Water Resources Planning and Management, ASCE. 2004.

International Conferences

- Liong, S. Y., Sivapragasam, C., Muttil, N., Doan, C. D., and **Yu, X. Y.** Efficient Water Management Techniques for Rapidly Urbanizing Countries. In Proceedings of Symposium on Innovative Approaches for Hydrology and Water Resources Management in the Monsoon Asia, University of Tokyo, pp. 71-78. 2001.
- **Yu**, **X. Y.**, Liong, S. Y. and Babovic, V. Hydrologic Forecasting with Support Vector Machine Combined with Chaos-inspired Approach. In Proceedings of 5th

International Conference on Hydroinformatics, Cardiff University, Cardiff, Wales, U.K., pp. 764-769. 2002.

- **Yu, X. Y.**, Liong, S. Y., and Babovic, V. An Approach Combining Chaos-Theoretic Approach and Support Vector Machine: Case Study in Hydrologic Forecasting. In Proceedings of the 13th APD-IAHR Congress, Singapore. pp. 690-695. 2002.
- **Yu, X. Y.** and Liong, S. Y. Forecasting of Chaotic Hydrological Time Series with Ridge Linear Regression in Feature Space. In Proceedings of 6th International Conference on Hydroinformatics, Singapore, pp. 1581-1588. 2004.
- Liong, S. Y., MD. Atiquzzaman and **Yu, X. Y.** Multi-objective Algorithm to Enhance Decision Making Process in Water Distribution Network Problems. In Proceedings of 2nd APHW Conference, Singapore, pp. 138-146.2004.
- **Yu, X. Y.** and Liong, S. Y. Enhanced Support Vector Machine for hydrological time series forecasting. 14th APD-IAHR Congress, 15 - 18, December 2004, Hong Kong (Accepted for publication).