1-1-2005

Support vector machines for environmental informatics : application to biological nitrogen removal in wastewater treatment plants

Yinghui Yang
Ryerson University

SUPPORT VECTOR MACHINES FOR ENVIRONMENTAL INFORMATICS: APPLICATION TO BIOLOGICAL NITROGEN REMOVAL IN WASTEWATER TREATMENT PLANTS

by

Yinghui Yang
Bachelor of Computer Science and Engineering
Beijing University of Aeronautics and Astronautics
Beijing, China 1998

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in the Program of
Electrical and Computer Engineering.

Toronto, Ontario, Canada, 2005

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Yinghui Yang

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Yinghui Yang

Instructions on Borrowers

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

SUPPORT VECTOR MACHINES FOR ENVIRONMENTAL INFORMATICS: APPLICATION TO BIOLOGICAL NITROGEN REMOVAL IN WASTEWATER TREATMENT PLANTS

Master of Applied Science, 2005

Yinghui Yang

Electrical and Computer Engineering, Ryerson University

Abstract

In order to meet the more stringent environmental regulations, the adaptive and optimal control strategies should be investigated for the biological nitrogen removal(BNR) processes in wastewater treatment plants. Because of the complex nature of the microbial metabolism involved, the conventional mechanistic models for nitrogen removal are difficult to formulate and the existing ones are still uncertain to some extent. Alternatively, the machine learning methods have been investigated as black-box modelling techniques. A new approach, Support Vector Machine (SVM) was proposed to be used to model the biological nitrogen removal processes in this thesis. Specifically, LS-SVM, a simplified formulation of SVM, was applied to predict the concentration of nitrate & nitrite (NO). The simulation results indicate that the proposed method has better generalization performance in comparison with generalized regression neural network, especially under weather conditions that are quite different from the training weather condition.

Acknowledgments

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

I am deeply indebted to my supervisor Prof. Dr. A. Guergachi from the school of ITM whose support, stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis.

I am very grateful for Dr.Gul Khan, my co-supervisor in Electrical and Computer Engineering department, who has been consistently giving me help and support in many aspects.

My colleagues from the Department of Electrical and Computer Engineering supported me in my research work. I want to thank them for all their help, support, interest and valuable hints. Especially I am obliged to J.Jiang, Y.Zhu, J.M.Yang, Y.J.Wang, Y.P.Li, M.Du, Z.Q.Lu. I also want to thank C.Wu-Tanenbaum and R. Smith for their assistance on the research server and computers.

The financial support from NSERC, CFI, OIT and Ryerson University is highly acknowledged. The helpful discussions with John B. Copp and Oliver Schraa in Hydromantis Inc. are also gratefully appreciated.

I would like to give my special thanks to my family whose love and support enabled me to complete this work. Especially my husband, Xuezhong, for his love that brings me joyful time along with my study.

Contents

List of Tables

List of Figures

ACRONYMS

BNR:	Biological Nitrogen Removal
BOD:	Biochemical Oxygen Demand
BPNN	Back-propagation Neural Network
COD:	Chemical Oxygen Demand
COST:	European Cooperation in the field of Scientific and Technical Research
ERM:	Empirical Risk Minimization
GRNN:	Generalized Regression Neural Network
GUI:	Graphical User Interface
IAWPRC:	International Association on Water Pollution Research and Control
IAWQ:	International Association on Water Quality
KLa:	Oxygen Transfer Coefficient
LS-SVM:	Least squares Support Vector Machines
ML:	Mixed Liquor
MLP:	Multi-Layer Perceptron
NARX:	Nonlinear AutoRegressive model with Exogenous Inputs
NN:	Neural Netwroks
NO:	Nitrate and Nitrite
NOE:	Nonlinear Output Error
RBFNN:	Radial Basis Function Neural Network
SRM:	Structural Risk Minimization
SVC:	Support Vector Classification
SVM:	Support Vector Machines
SVR:	Support Vector Regression
TKN:	Total Kjeldahl Nitrogen (organic N + NH_4^+)
TN:	Total Nitrogen
TSS:	Total Suspended Solids

NOTATIONS

N	the number of training data
φ	feature function
x	input vector
x_k	input value
k	index of the input and output data point
$\hat{y}$	the predicted output value
ω	weight vector in linear function
b	bias term in linear function
f	target function
h	VC dimension
y	output vector
y_k	output value
$\Re_{emp}$	empirical error
$\Re$	generalization error
ε	the width of the band around the target function
L	Lagrangian expression
J_P	primal cost function
J_D	dual cost function
α_k, α_k^*	Lagrangian multipliers
ξ_k, ξ_k^*	slack variables
C	regularization parameter in ε-insensitive SVM cost function
γ	regularization parameter in LS-SVM cost function
σ	kernel parameter in RBF kernel function
e_i	the error at each data point with index i

Chapter 1

Introduction

1.1 Motivation

As the regulations for effluent quality are getting more and more stringent throughout North America, the advanced biological wastewater treatment (WWT) techniques are required to achieve better level of nutrient removal. Such techniques include conversion of ammonia nitrogen to nitrate by biological nitrification and removal of nitrate by biological denitrification. To accommodate plant influent fluctuations and other disturbances, there is a need to investigate the development and implementation of adaptive process control strategies, so that more precise and timely controls are achieved for the aforementioned techniques. As the basis of the development of the adaptive controllers, estimating the dynamics of the concentrations of some important trace elements (e.g., nitrogen) in the effluent is of primary consideration.

Conventionally, mechanistic models have been the most commonly used method for predicting the process dynamics so as to estimate the concentrations of various elements. Many mechanistic models and various control strategies have been incorporated in different software packages to address practical wastewater treatment problems. However, mechanistic models suffer from several fundamental deficiencies that have been discussed extensively by several authors in the literature [2, 3, 21, 32]. A recent synthesis and consolidation of these deficiencies can be found in the article by Guergachi and Patry [12]. The conclusion can be summarized as: mechanistic models are not able to deal with uncertainty in an effective manner. Mechanistic models can only roughly approximate the underlying function. The model

parameters are calibrated using empirical method, such as using the real data. That means, the mechanistic model with specific parameters can only reflect a portion of the dynamics of the complex wastewater treatment systems. The data used to determine the parameter have an impact on the model performance. Furthermore, the numerous parameters (e.g., kinetic and stoichiometric parameters) involved in the mechanistic models make the models too complex to use.

An alternative approach to modelling the uncertainty in wastewater treatment systems is machine learning. Machine learning is a black-box modelling technique that make use of empirical data. A lot of research work has been done to investigate the application of neural networks (NN) in modelling the wastewater treatment processes. However, as most of the empirical models, the weakness of neural network is that its use is limited to the range of data over which it was trained. The generalization ability of neural network is not so satisfactory. While we are aware of the strengths and weaknesses of many of the exiting technologies such as neural networks, fuzzy logic and knowledge-based systems, a novel system modelling approach called support vector machine (SVM) [49] emerged recently as a natural consequence of the results of statistical learning theory and needs extensive investigation at the empirical level.

The formulation of SVM is based on Vapnik's statistical learning theory (SLT) and structural risk minimization (SRM) [48]. According to the statistical learning theory, an upper bound on the generalization error guarantees the existence of a hypothesis with a certain level of accuracy. This generalization bound considered the trade-off between the empirical error and complexity of the model. By applying the structural risk minimization, an appropriate hypothesis can be found with proper complexity to fit the given data set.

Several researchers and authors have reported that this approach, when it is applied to pattern recognition, the discrete case, is a powerful one and delivers better results than the other traditional and competing approaches such as neural networks do [41, 47]. There is, however, a need to carry out more investigations (empirical first and then theoretical) of the performance of the support vector machines (SVMs) in the case of *continuous* and *complex* systems such as biological wastewater treatment plants. It is the intention of this thesis to present an empirical investigation of SVMs by applying them to the modelling of nitrogen

removal in wastewater treatment systems.

After the SVM concept was introduced by Vapnik, many variations of SVMs have been developed by other researchers to leverage the strength while overcoming the difficulties in applications of the initial SVM concept. One of these variations known as the least squares SVM (LS-SVM) was introduced by Suykens and co-workers [45]. It is this variation that will be investigated in this study. An advantage of the LS-SVM is its simplicity in terms of memory requirements and algorithmic implementation, and the fact that LS-SVM can be used for cases where adaptive and online learning is needed.

In modelling the nitrogen removal processes and predicting a certain variable in wastewater treatment systems, the output depends on many inputs and control variables, as well as on the previous values of the output itself. Because of the correlation between the variable to be predicted and its previous values, as well as with the current and previous values of the other variables, NARX (Nonlinear AutoRegressive model with Exogenous Inputs) modelling concept was integrated into LS-SVM to accommodate the information contained in both the current and historical data. Extensive simulations using simulated data generated from the wastewater treatment software package GPS-X [10] were carried out to examine how LS-SVM can perform on predicting nitrogen concentrations in treated wastewater. Specifically, LS-SVM Matlab toolbox is used in the estimation of nitrate & nitrite (NO) concentration. Moreover, to compare the performance of LS-SVM in prediction, the simulations using a special neural network called Generalized Regression Neural Network (GRNN) under the same settings and procedures were also implemented. The simulation results indicate that LS-SVM has better generalization ability than GRNN because it can predict the response of the system to dynamic variations while only trained on a limited set of data that don't include all of the patterns of the variation. Through the sensitivity analysis, it can be seen that the performance of LS-SVM is highly dependent on the selection of NARX memory orders.

The results of this study confirmed the advantages of LS-SVM in modelling uncertainty of complex continuous system. Also, some future work directions can be derived from this empirical study.

1.2 Summary of Contributions

Although SVMs have had plenty of applications since its emergence in 1995, the research of SVM mostly focused on classification or pattern recognition, which are discrete cases. Based on our literature review, there were very few paper discussing the application of SVM in complex continuous system modeling. In wastewater treatment system, especially the biological nitrogen removal, there hasn't been any research that investigated SVMs as the modelling method. In contrast, neural networks have been applied in wastewater treatment processes modelling in a wide range, including some research on nitrogen removal. This thesis contributed to this area in the following aspects:

This research is innovative to apply SVM to complex continuous system modelling. For this purpose, a variety of SVM-based methods were investigated and LS-SVM was selected as a better one for modeling the dynamic of complex continuous system.

In the case of modelling biological nitrogen removal process, the domain knowledge has been studied and relevant input and output variables were identified, in order to enable the modelling procedure and improve the simulation performance.

A critical step leading to the successful results is the selection of COST (European Co-operation in the field of Scientific and Technical Research) simulation benchmark in GPS-X to generate the simulated data for LS-SVM training and testing.

To take advantage of both regression and time series prediction ability of LS-SVM, NARX model was proposed to transform the input and output variables into a new space with embedded memory. Using this new input and output space enables accurate prediction of the target variable, which can't be obtained by using regression or time series prediction individually.

All of the learning and prediction simulations, including the input-output transformation, parameter tuning, training and prediction, were implemented in MATLAB with LS-SVM Toolbox. The simulation results were visualized in tables and plots to make it interpretable.

The comparison between the prediction performance of LS-SVM and GRNN indicated that LS-SVM had better generalization performance than GRNN when tested over the data not from the training data set. These empirical results reinforce the power of statistical

learning theory and SRM principle.

1.3 Thesis outline

This thesis is organized into 5 chapters. Besides this introduction chapter, chapter 2 presents the background of this study, the previous work in this area and the proposed approach. Chapter 3 reviews the fundamental concepts, algorithms and related tools of SVM. Three kinds of neural network, BPNN, RBFNN and GRNN, are illustrated and compared. LS-SVM, NARX and NOE prediction models are introduced and proposed as the techniques for solving the target problem. In chapter 4, the simulation procedures and results are illustrated. The conclusion and the future work are summarized in chapter 5.

Chapter 2

Literature Review

2.1 Biological Nutrient Removal

The municipal wastewater includes significant nutrients which need to be removed. Nitrogen and phosphorus are the principal nutrients of concern in treated wastewater discharges. In general, the nitrogen and phosphorus contained in the discharged wastewater, if not removed, will stimulate the growth of algae and rooted aquatic plants in shallow streams. As the consequence, eutrophication phenomenon will be accelerated in the lake and reservoirs, which not only has an impact on the aesthetics of the water body but also interfere with the beneficial use of the water resources, particularly when they are used for water supplies, fish propagation, and recreation. If the nitrogen concentrations are significant in treated effluents, other adverse effects may be caused. Toxicity toward aquatic life and hazard to public health will be presented so that the wastewater can not be reused. Therefore, the control of nitrogen and phosphorus is becoming increasingly important in water quality management and in the design of wastewater treatment plants [35].

The regulations of effluent quality with regard to these two chemicals are getting more and more stringent throughout North America. As the result, the discharge of one or both of these constituents have to be controlled.

The traditional wastewater treatment plant placed emphasis on eliminating carbonaceous pollutants and sludge. A typical traditional wastewater treatment plant consists of a primary biological reactor that remove the carbonaceous organic and a secondary settler that clarify the suspended solids. Advanced wastewater treatment is defined as the additional treatment

needed to remove suspended and dissolved substances remaining after conventional secondary treatment. Since the early 1970s, the number of advanced wastewater treatment facilities has increased significantly, and a great deal of information has been published, especially with respect to the removal of nitrogen and phosphorus [35].

Various treatment methods have been used employing chemical, physical, and biological systems to limit or control the amount and form of nitrogen discharged by the treatment system. Biological nutrient removal is relatively low-cost means of removing nitrogen and phosphorus from wastewater. Recent experience has shown that biological processes are reliable and effective in removing nitrogen and phosphorus [35].

Nutrient removal options that need to be considered include the following:

1. Nitrogen removal without phosphorus removal
2. Nitrogen and phosphorus removal
3. Phosphorus removal with or without nitrification
4. Year-round removal of phosphorus with seasonal removal of nitrogen

In our study, we focus on the first option, nitrogen removal without phosphorus removal.

The two principal mechanisms for the removal of nitrogen are assimilation and nitrification/denitrification.

Nitrogen in untreated wastewater is principally in the form of ammonia or organic nitrogen, both soluble and particulate. Untreated wastewater usually contains little or no nitrite or nitrate. During biological treatment, most of the particulate organic nitrogen is transformed to ammonium and other inorganic forms. A portion of the ammonium is assimilated into the cell material of the biomass. Most of the nitrogen in treated secondary effluent is in the ammonium form. In order to remove ammonia, the advanced biological nitrogen removal (BNR) techniques, such as nitrification-denitrification, are required in addition to the traditional aeration tank and secondary settler.

In nitrification-denitrification, the removal of ammonia (NH_4^+) is accomplished in two conversion steps. These two processes are considered separately. In the first step, nitrification, ammonium is converted to nitrite (NO_2^-) and nitrate (NO_3^-) by autotrophic bacteria

under aerobic conditions. However, the nitrogen has merely changed forms and not been removed. In the second step, denitrification, nitrate and nitrite are converted to a gaseous product by heterotrophic bacteria under anoxic conditions with the use of organic compounds as reducing agent. The gaseous product, N_2, is one of the components of the atmosphere and can be directly released to the air. Anoxic zones can be placed either in the beginning of the bioreactor (pre-denitrification) or in the end of the bioreactor (post-denitrification). In a pre-denitrifying system, an internal recirculation flow is usually introduced to transport the nitrate rich water back to the anoxic zone. The nitrogen transformation within the BNR process is illustrated in Figure 2.1.

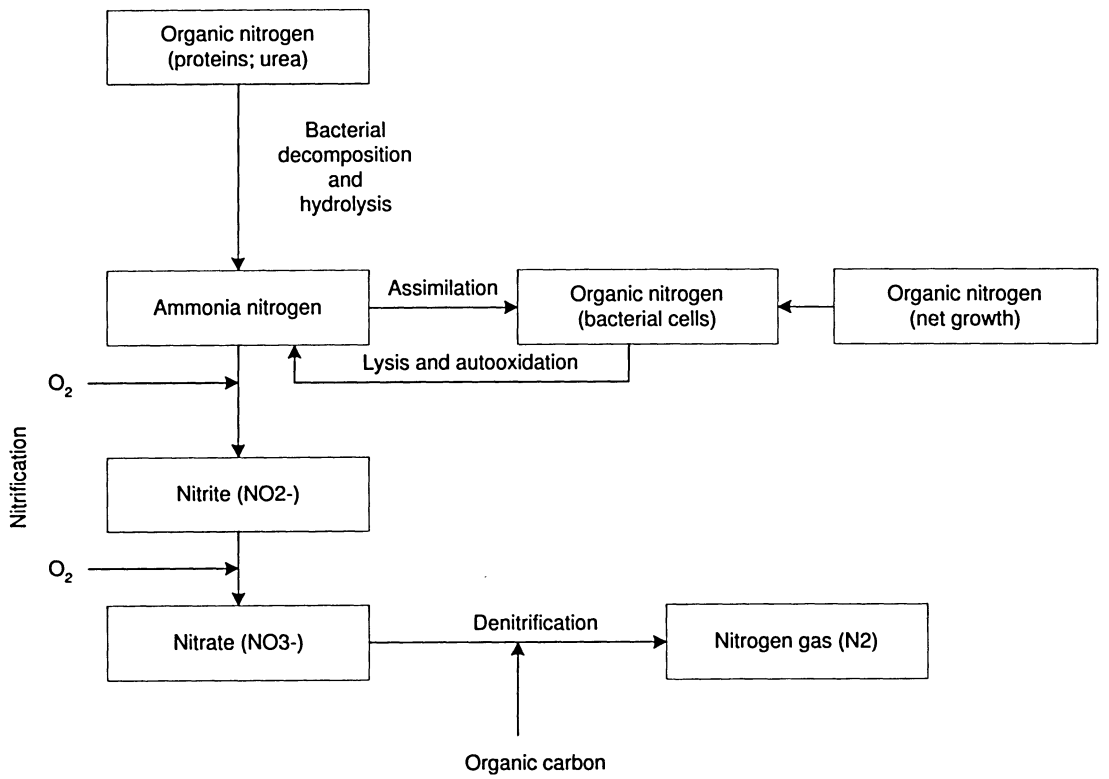


Figure 2.1: Nitrogen transformations in biological treatment processes [35].

The reaction of nitrification can be illustrated by the equations:

$$55NH_4^+ + 76O_2 + 109HCO_3^- \rightarrow C_5H_7O_2N + 54NO_2^- + 57H_2O + 104H_2CO_2 \quad (2.1)$$

$$400NO_2^- + NH_4^+ + 4H_2CO_3 + HCO_3^- + 195O_2 \rightarrow C_5H_7O_2N + 3H_2O + 400NO_3^- \quad (2.2)$$

The denitrification can be described as:

$$NO_3^- \rightarrow NO_2^- \rightarrow NO \rightarrow N_2O \rightarrow N_2 \quad (2.3)$$

The biological nitrogen removal can be implemented within the traditional wastewater treatment plant by modifying the biological conditions within the plant. For example, Activated sludge process (ASP) is one of the most widespread processes for biological wastewater treatment. ASP has undergone successive modifications since its original application, which have improved its efficiency and expanded its use from the elimination of carbonaceous organic matter to the simultaneous removal of nutrients, especially nitrogen (as nitrate and ammonia). We can see later about the application of ASP in the data collection part.

According to a study [35], less than 30 percent of the total nitrogen is removed by conventional secondary treatment. Biological nitrification can remove 5-20% of the total nitrogen entering process, while denitrification can remove 70-95% of the total nitrogen, provided that the proper environment is produced and controlled effectively. In addition, biological treatment is relatively low cost. As the result, biological nitrification/denitrification is becoming more and more important in nutrient removal in wastewater treatment.

Even though the application of biological nitrogen removal has many benefits, we should be aware of the difficulties in utilizing this technique. Nitrogen can occur in many forms in wastewater and undergo numerous transformations in wastewater treatment. The transformations allow the conversion of ammonia-nitrogen to products that can easily be removed from the wastewater. Because of the complicated activities of microbial metabolism involved, nitrogen removal is a nonlinear, dynamic, and time variant complex process. To properly control such a complex system and achieve the highest level of nitrogen removal, the function between a certain target variable in the effluent and the affecting variables in the influent needs to be established.

There is a need to investigate the development and implementation of adaptive process control strategies accommodating influent fluctuation and other disturbances, in order to obtain more precise and timely controls. The first consideration of developing the adaptive

controllers is to estimate the concentrations of some important trace elements in effluent in respond to the dynamic change of the influent characteristics.

The target problem of this thesis is to predict the concentration of an important indicator variable in nitrogen removal processes, Nitrate & Nitrite (NO), in a short term. From Figure 2.1, we can see the concentration of Nitrate and Nitrite indicates the effect of nitrification/denitrification. The objective is to study how the concentration of NO is affected by the influent fluctuations and some control strategies. The purpose is to examine the effectiveness of our proposed method on a benchmark biological wastewater treatment process involving nitrification/denitrification. We will further apply the proposed method to real systems if the simulation results are appealing.

2.2 Previous work

To reach the goal of predicting the NO concentration in the treated effluent, it is necessary to establish a model which can map the target output to the selected inputs. As part of the study, various methodologies for modelling and simulating the dynamics of nitrogen removal are studied. Models are mathematical and computer representation of real systems. The real systems of interest here are the unit processes in a wastewater treatment plant, such as the aeration tank or the sedimentation tank in an activated sludge treatment system, or the combination of them. There are two types of models which are widely applied in wastewater treatment area, mechanistic models and empirical models.

Mechanistic models (Figure 2.2) are based on first principles, that is, laws of physics, chemistry, and biology. The models are based on theoretical results and the model parameters often have physical significance. Actual data can be used to estimate the model parameters. The aim of parameter estimation is not only to fit the model to the data but also calculate the parameter values that are good estimates of the true values of the physical quantities. For example, the law of conservation of mass is the essential building block on which dynamic equations are based. Data collected in the real system are used to adjust rate constants and other parameters in the equations. This is a kind of "bottom-up" approach to modelling in which the model is provided a firm foundation by natural laws. These laws

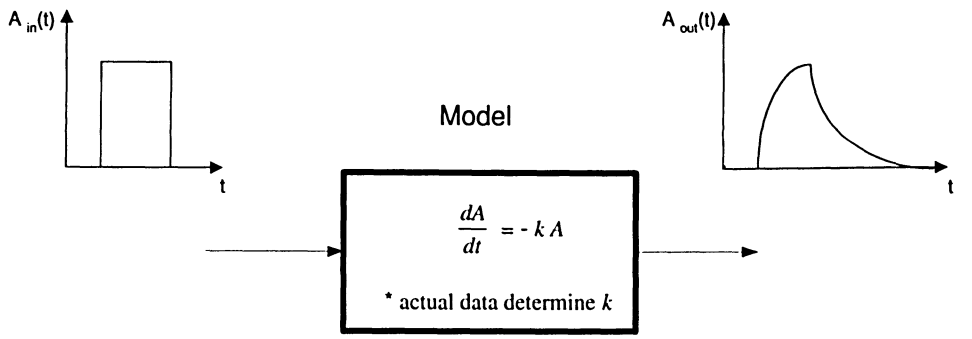


Figure 2.2: Mechanistic Models are Based on First Principles

help us to understand more thoroughly the behaviour observed in the real system and the model.

Empirical models rely solely on data. The choice of the structure of the empirical model is more arbitrary. The statistical properties of the parameter may not be as meaningful in physical sense as in mechanistic models. In this approach, the model structure is determined by selecting from a set of general mathematical functions such as those shown in the center of Figure 2.3. Essentially, the data "speak for themselves" and ultimately dictate the form of equations - selected among a set of candidate equations on the basis of goodness-of-fit to be used in the model. This "top-down" approach to modelling is deemed weaker than the mechanistic approach as the model structure is simpler and the meaning less clear. A different set of data can often result in a different model structure and we are not always certain how to interpret the meaning of parameters in the model. Nevertheless, empirical models are useful as long as their use is limited to the data ranges over which they are fitted, and in many cases are the only choice when we have limited knowledge about the real system being modelled. For example, empirical models often give good predictions and can ultimately lead to the discovery of more general mechanistic expressions.

Conventionally, structured process models (mechanistic models) have been the most commonly used method for simulating and predicting the wastewater treatment process dynamics. Such models are developed on the basis of the information that is available about the physical process mechanisms. For instance, the series of Activated Sludge Models, i.e. ASM No.1,2,3 [17, 18, 20] developed by IAWPRC (International Association on Water Pollution

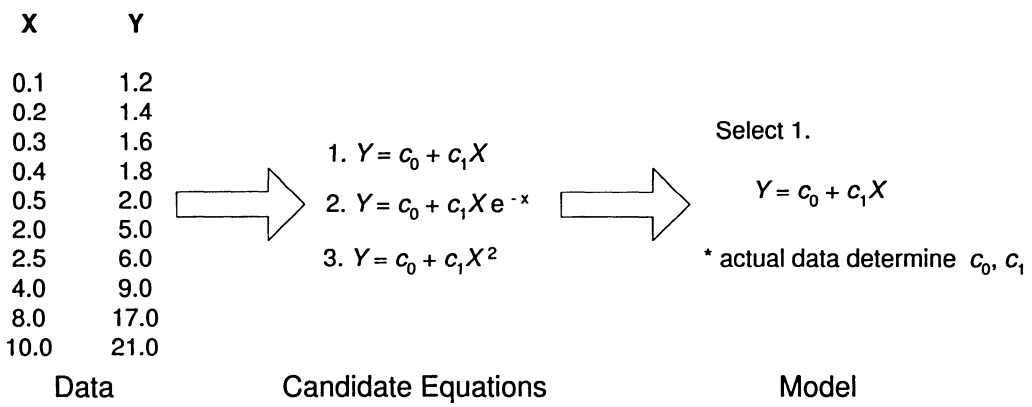


Figure 2.3: Empirical Models Fit the Data to a Certain Model Form

Research and Control) and IAWQ (International Association on Water Quality), predict oxygen consumption, sludge production, nitrification and denitrification of activated sludge systems, by mathematically manipulating the kinetic and stoichiometric equations of the reactions occurring in the bioreactor. However, establishing such mechanistic models is a formidable task for biological nutrient removal processes. A multitude of microbial reactions coupled with environmental interactions that are nonlinear, time-variable and still uncertain are actively involved in these processes [27]. The complexity of such mechanistic models of biological WWT processes is unparalleled in the chemical industry [21]. Beck [2] pointed out the lack of identifiability of the IAWPRC Model. "The lack of model identifiability is due to the fact that available models explain just a portion of the behaviour of highly complex systems. The other portion which is not accounted for by those models shows itself through the variability of model parameters. Lack of identifiability is then an inherent feature of complex systems" [12]. Kang-Young Ko et al [25] claimed that these models are too complicated due to many stoichiometric and kinetic parameters that require extensive off-line calibration and long computation time. Moreover, the resulting analytic dynamic models are incompletely formulated due to biochemical nature and numerous parameters of wastewater treatment process and they are only valid for municipal wastewater treatment processes.

As both mechanistic models and empirical models are not quite suitable in modelling wastewater treatment processes, researchers have been seeking more effective modelling techniques. In the artificial intelligence literature, an emerging modelling approach is machine

learning. Machine learning method is similar to empirical models as it is based on a set of data examples. The advantage of using machine learning methods is one doesn't need to know the underlying probability distribution of the data. Most of the machine learning methods can approximate any functions (linear or nonlinear) to an arbitrary degree of accuracy over a compact interval.

For machine learning methods, the model identifiability is not essential. Neural networks (NN), for instance, "are fundamentally non-identifiable, yet they have been used extensively and successfully in the area of wastewater treatment process modelling" [12]. NN models have already been applied to the implementation of "software sensors" [27, 51] and to the estimation of wastewater parameters[7, 14]. The development of NN models for the model-based dynamic control of the entire wastewater treatment process can be found in [15, 11, 43].

Several other types of empirical models have been developed and applied for the wastewater treatment processes. Fuzzy control is used to reduce the effect of unmeasured disturbances [37]. Fuzzy inference method is applied to the fuzzy model to predict a possibility distribution of a future BOD (Biochemical Oxygen Demand) value and take appropriate judgement and control based on the prediction[42]. Time series technique is used for parameter estimation and process optimization in GPS-X [10]. GPS-X has both mechanistic and empirical models. Note that the difference between empirical and mechanistic models is more a continuum than a crisp classification as mechanistic models often have some empirical component. As mentioned before, actual data are used to modify parameters in the structured models.

The problem of the aforementioned empirical modelling techniques is that the model is limited over the range of training data set and there is no guarantee that the model is correct on the data outside the training data set. Another problem is the difficulty to determine the complexity of the model. "If the size of the data set used for model identification is small while the number of model parameters is large (i.e. the model is highly complex), then the problem of data overfitting by the model would occur. On the other hand, if the model is too simple and, therefore, the number of parameters is sufficiently low compared to the size of the data set, then the explanatory power of the model would be so low that the value of the objective function would become very high, meaning the model prediction of the true

process behaviour is of a low quality. Consequetnly, the degree of complexity of a process model has to be adjusted to the amount of data that is available for the identification of this model. For any fixed amount of data, there is an optimal mdoel complexity that has to be determined. Models that are more complex would cause overfitting, and models that are less complex would lead to a low prediciton quality" [12]. The mathematical framework that was developed in Guergachi and Patry [12] defines all the necessary concepts and tools that help determine the optimal structure complexity of a WWT process model, corresponding to a fixed amount of data.

2.3 Proposed method

While we are aware of the strengths and weaknesses of many of the exiting technologies such as neural networks, fuzzy logic and time series, the introduction of an innovative machine learning approach by Vapnik [49] provided answers for the shortcomings. Support Vector Machine emerged as a natural consequence of the results of statistical learning theory and structural risk minimization principle. Statistical learning theory provided theoretical foundation of the generalization ability of SVM. Structural risk minimization offers a structured way to determine the complexity of the model so as to avoid overfitting. The solution of SVM has the properties such as global optimum, sparseness and bounded generalization error.

Even though the theoretical foundation of SVM is very strong, there is, however, a need to carry out more investigations (empirical and theoretical) of the performance of SVMs. Since its introduction, SVM has been applied to many classification problems and outperformed other techniques. However, in the case of continuous and complex systems such as biological wastewater treatment plants, there were much less research that has been done. It is the intention of this thesis to present an empirical investigation of SVMs by applying them to the modelling of BNR in wastewater treatment systems.

Some variations of SVMs have been developed by researchers to leverage the strength while overcome some difficulties in applications of standard SVM. For example, SVM solutions are usually found via batch algorithms and can not be used for online learning. A simplication of SVM formulation known as the least squares SVM (LS-SVM) has nice prop-

erties in the sense that their solution can be found by solving a set of linear equations making the algorithm amenable for on-line application.

For these reasons, in our study, we proposed to make use of LS-SVM to model the dynamics of the nitrogen removal processes and predict the NO concentration under different weather conditions. In order to compare the performance of our proposed methods with NN style methods, one special neural network, generalized regress neural network, was also applied to the modelling and prediction as a comparison to LS-SVM.

Chapter 3

Theoretical Background and Proposed Solution

Mathematical models have been developed to solve a variety of problems such as classification and regression. However, the nature of the systems that scientists and engineers deal with has evolved. Two characteristics of such systems have emerged as a result of this evolvement: increased complexity and increased amount of data. More and more tasks with high complexity cannot be solved by classical programming techniques, since no mathematical model of the problem is available. One solution for such complex system modeling, analysis and control is to make use of the second characteristic (increased data) to address the challenges posed by the first characteristic (increased complexity). The construction of machines capable of learning from experience (historical data) has received an enormous impetus from the advent of electronic computers. Machine learning methodology is an artificial intelligence approach to develop and train a model to recognize the pattern or underlying mapping of a system based on a set of training examples consisting of input and output patterns. It has been demonstrated that machines can display a significant level of learning ability, though the boundaries of this ability are far from being clearly defined.

In this chapter, we will give an overview of the important concepts of one important machine learning methodology — supervised learning. Then we will introduce a family of innovative supervised learning methods - Support Vector Machines (SVMs) and explain why Support vector machines meet many of the challenges confronting machine learning systems, in comparison to neural networks. MLP, RBF, and GRNN will be emphasized when neural networks (NNs) are introduced. Three prediction models will be explained. At the end, a

solution combining LS-SVM, NARX and NOE prediction models is proposed for the target problem.

3.1 Supervised Learning

There are many situations that the traditional programming approach cannot be effective. Such as, the system designer cannot precisely specify the method by which the correct output can be computed from the input data, or the computation may be very expensive. When such situations arise, an alternative strategy for solving this type of problem is for the computer to attempt to learn the input/output relationships from examples. The approach of using examples to synthesize models is known as learning, and in the particular case when the examples are input/output pairs it is called supervised learning. It is inspired by the fact that the kids are taught by the teacher to learn the real world problems such as classification by looking at some examples and generalizing to new items. The examples of input/output are referred to as the training data. The input/output pairings typically reflect a functional relationship mapping inputs to outputs, though this is not always the case as for example when the outputs are corrupted by noise. When an underlying function from inputs to outputs exists, it is referred to as the target function. The estimate of the target function which is learnt or output by the learning algorithm is known as the solution of the learning problem. The solution is chosen from a set of candidate functions which map from the input space to the output domain. Usually we will choose a particular set or class of candidate functions known as hypotheses before we begin trying to learn the correct function. Hence, the choice of the set of hypotheses (or hypothesis space) is one of the key ingredients of the learning strategy. The second important ingredient is the algorithm which takes the training data as input and selects a hypothesis from the hypothesis space. It is referred to as the learning algorithm. The ability of a hypothesis to correctly perform input/output mapping outside the training set is known as its generalization, and it is this property that shall be optimized. The aim is to find the hypothesis that gives the right output according to the generalization criterion. The hypothesis has become a functional measure rather than a descriptional one, that is, it is not necessary to be a correct representation of the true function. In this sense the criterion places no constraints on the size or on the 'meaning' of

the hypotheses — for the time being these can be considered to be arbitrary.

The learning models are divided into two types according to the way in which the training data are generated and how they are presented to the learner. One type is batch learning in which all the data are given to the learner at the start of learning. The other type is online learning in which the learner receives one example at a time and the hypothesis is updated according to each new example. The advantages of online learning is it can be used when there is no fixed training set (new data keeps coming in) and it is better at tracking nonstationary environments (where the best model gradually changes over time).

Support vector machines, the learning approach we will emphasize in the following of this chapter, cannot be used online. But a simplified SVM method — LS-SVM can be used online. Our target problem is stochastic, so online learning is more suitable. But in a short term, we can treat it as a stationary process so that batch learning can be applied. The study of this thesis concerns the supervised learning methodology from batch training data.

The advantages of machine learning methodology are very attractive. Firstly, the range of applications that can potentially be solved by such an approach is very large. Secondly, it appears that we can also avoid much of the laborious design and programming inherent to the traditional solution development methodology, at the expense of collecting some labelled data and running an off-the-shelf algorithm for learning the input/output mapping. Finally, there is the attraction of discovering insights into the way that humans' brains function, an attraction that inspired early work in neural networks.

There are, however, many difficulties inherent to the learning methodology that deserve careful study and analysis. The first is that the learning algorithm may prove inefficient as for example in the case of local minima. The second is that the size of the output hypotheses can frequently become very large and impractical. The third problem is that if there are only a limited number of training examples, too rich a hypothesis class will lead to overfitting and hence poor generalization. The fourth problem is that frequently the learning algorithm is controlled by a large number of parameters that are often chosen by tuning heuristics, making the system difficult and unreliable to use.

Take neural networks (NN) as example, despite the notable successes of NN in many kinds of applications, neural networks often suffer from the local minima, overfitting and a large

number of parameters. In addition, there is frequently a lack of understanding about the conditions that render an application successful, in both algorithmic and statistical inference terms. We will see in the next section that Support Vector machines address all of these problems.

The most common tasks in supervised learning problems belong to two major categories: classification and regression. Time series prediction is another often confronted task that can be dealt with as a regression problem. In the sequel of this thesis, we will focus on regression and time series prediction.

3.2 Support Vector Regression

3.2.1 Vapnik's ε-insensitive SVM

Support Vector machines (SVM) are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm derived from optimization theory and implements a learning bias derived from statistical learning theory. This learning strategy introduced by Vapnik and co-workers [48, 49] is a principled and very powerful method that, in the last few years since its introduction, has already outperformed most other systems in wide variety of applications, especially in pattern recognition [41, 47].

The simplest support vector machines were developed for binary classification. With continuous extension and advancement, SVMs were applied to functional approximation and time series prediction.

Linear learning machines are the fundamental formulations of SVMs. The objective of the linear learning machine is to find the linear function that minimizes the generalization error from a set of functions which can approximate the underlying mapping between the input and output data. Consider the following set of linear functions:

$$f(x;\omega,b) = \omega^T x + b \tag{3.1}$$

with N given training data that have input values $x_k \in \Re^n$ and output values $y_k \in \Re$. ω is the weight vector and b is the bias.

The empirical error is defined as

$$\Re_{emp}(\omega, b) = \frac{1}{N}\sum_{k=1}^{N}|y_k - f(x;\omega,b)| = \frac{1}{N}\sum_{k=1}^{N}|y_k - \omega^T x - b| \tag{3.2}$$

The generalization error can be expressed as

$$\Re(\omega, b) = \int |y - f(x;\omega,b)|p(x,y)dxdy \tag{3.3}$$

which measures the error for all input/output patterns that would be generated from the underlying generator of the data characterized by the probability distribution $p(x, y)$. Unfortunately, the probability distribution $p(x, y)$ is not known in practice.

According to statistical learning theory [49], the generalization error can be upper bounded in terms of training error and a confidence term as shown in Equation 3.4:

$$\Re(\omega, b) \leq \Re_{emp}(\omega, b) + \sqrt{\frac{h(\ln(2N/h)+1) - \ln(\eta/4)}{N}} \tag{3.4}$$

where N is the number of training examples and h is VC dimension. VC dimension is used to describe the complexity of the learning system [49]. This bound holds with probability $1 - \eta$. The term on left side represents generalization error. The first term on right side is empirical error calculated from the training data and the second term is called confidence term which is associated with the VC dimension h of the learning machine. The relationship between these three items is illustrated in Figure 3.1.

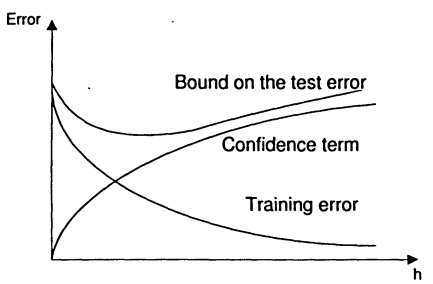


Figure 3.1: Upper bounded generalization error

Thus, even though we don't know the underlying probability distribution based on which the data are generated, it is possible to minimize the upper bound of the generalization

error in place of minimizing the generalization error itself. That means one can minimize the expression in the right side of the equation 3.4.

Unlike the principle of Empirical Risk Minimization (ERM) which aims to minimize the training error and is used in neural network training, SVMs implemented Structural Risk Minimization (SRM) in their formulations. SRM principle takes both the training error and the complexity of the model into account, and intends to find the minimum of the sum of these two terms as a trade-off solution (as shown in Figure 3.1) by searching a nested set of functions of increasing complexity [48, 49].

Vapnik's formulation of SVM regression employed a so-called Vapnik's ε-insensitive cost function defined as

$$|y - f(x)|_\varepsilon = \begin{cases} 0, & if\ |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon, & otherwise \end{cases} \tag{3.5}$$

Estimation of the linear function in 3.1 needs to find ω and b. ω can be optimized by replacing f(x) in Equation 3.5 with the linear function in Equation 3.1 and solving a constrained optimization problem shown in formulation 3.6:

$$\min_{\omega,b,\xi,\xi^*} J_P(\omega, \xi, \xi_k^*) = \frac{1}{2}\omega^T\omega + C\sum_{k=1}^{N}(\xi_k + \xi_k^*) \tag{3.6}$$

$$\begin{aligned} \textit{such that} \quad & y_k - \omega^T x_k - b \leq \varepsilon + \xi_k, k = 1, \ldots, N \\ & \omega^T x_k + b - y_k \leq \varepsilon + \xi_k^*, k = 1, \ldots, N \\ & \xi_k, \xi_k^* \geq 0, k = 1, \ldots, N. \end{aligned}$$

C is a regularization parameter which adjusts the balance of the training error and the complexity of the system. This formulation is called Vapnik's ε-insensitive cost function due to ignoring the error within a band around the target function. ξ_k and ξ_k^* are slack variables. ω is the weight vector of the linear function in Equation 3.1 and is derived from the solution of the optimization problem in Equation 3.6. b can be calculated using the complementary conditions [45]. This formulation 3.6 is a constrained optimization problem in primal weight space. The Lagrangian for this problem is:

$$
\begin{aligned}
L(\omega, b, \xi_k, \xi_k^*; \alpha, \alpha^*, \eta, \eta^*) &= \frac{1}{2}\omega^T\omega + c\sum_{k=1}^{N}(\xi_k + \xi_k^*) \\
&\quad - \sum_{k=1}^{N}\alpha_k(\varepsilon + \xi_k - y_k + \omega^T x_k + b) \\
&\quad - \sum_{k=1}^{N}\alpha_k^*(\varepsilon + \xi_k^* + y_k - \omega^T x_k - b) \\
&\quad - \sum_{k=1}^{N}(\eta_k\xi_k + \eta_k^*\xi_k^*)
\end{aligned} \tag{3.7}
$$

with positive Lagrange multipliers $\alpha_k, \alpha_k^*, \eta_k, \eta_k^*$. The conditions for optimality are

$$
\begin{cases}
\frac{\partial L}{\partial \omega} = 0 \;\rightarrow\; \omega = \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)x_k \\
\frac{\partial L}{\partial b} = 0 \;\rightarrow\; \sum_{k=1}^{N}(\alpha_k - \alpha_k^*) = 0 \\
\frac{\partial L}{\partial \xi_k} = 0 \;\rightarrow\; c - \alpha_k - \eta_k = 0 \\
\frac{\partial L}{\partial \xi_k^*} = 0 \;\rightarrow\; c - \alpha_k^* - \eta_k^* = 0.
\end{cases} \tag{3.8}
$$

Replacing 3.8 back to 3.6, the dual problem is a QP problem described in the formulation 3.9:

$$
\begin{aligned}
\min_{\alpha, \alpha^*} J_D(\alpha_k, \alpha_k^*) &= -\frac{1}{2}\sum_{k,l=1}^{N}(\alpha_k - \alpha_k^*)(\alpha_l - \alpha_l^*)x_k^T x_l \\
&\quad -\varepsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + \sum_{k=1}^{N} y_k(\alpha_k - \alpha_k^*) \\
\textit{such that} \quad & \sum_{k=1}^{N}(\alpha_k - \alpha_k^*) = 0 \\
& \alpha_k, \alpha_k^* \in [o, c].
\end{aligned} \tag{3.9}
$$

Lagrangian multipliers α_k and α_k^* can be obtained by solving this dual optimization problem. The examples (x_k, y_k) corresponding to the non-zero Lagrangian multipliers α_k and α_k^* are called support vectors. Only support vectors are involved in the expression of the target function as in Equation 3.10:

$$f(x) = \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)x_k^T x + b \tag{3.10}$$

The advantage of using the dual representation is derived from the fact that in this representation the number of free parameters relies on the number of support vectors instead of the number of dimensions of the input space (equals the dimension of weight vector in the primal space). This property enables the regression in a high dimensional space.

However, most of the practical problems are nonlinear instead of simple linear ones. Kernel functions extended the power of linear learning machine by mapping the input data into a high dimensional feature space. A linear learning machine can be employed in the feature space to solve the original non-linear problem. Kernel functions satisfying Mercer condition [34] not only enable implicit mapping of data from input space to feature space but also ensure the convexity of the cost function which leads to the unique optimum. Mercer condition states that a continuous symmetric function $K(x, z)$ must be positive semi-definite to be a kernel function which can be written as inner product between the data pairs as in Equation 3.11 [45]:

$$K(x, z) = \sum_{i=1}^{n_\aleph} \lambda_i \phi_i(x)\phi_i(z) = \varphi(x)^T \varphi(z) \tag{3.11}$$

where $n_\aleph$ is the dimension of the feature space $\aleph$ and λ_i are positive numbers.

Several typical choices of kernels are linear, polynomial, MLP and RBF kernel. Their expressions are as following:

$$K(x, x_k) = x_k^T x \quad (\textit{linear kernel}) \tag{3.12}$$

$$K(x, x_k) = (\tau + x_k^T x)^d \quad (\textit{polynomial kernel}) \tag{3.13}$$

$$K(x, x_k) = \exp(-\|x - x_k\|_2^2/\sigma^2) \quad (\textit{RBF kernel}) \tag{3.14}$$

$$K(x, x_k) = \tanh(\kappa_1 x_k^T x + \kappa_2) \quad (\textit{MLP kernel}) \tag{3.15}$$

In the Lagrangian expression, the training examples never appear isolated but always in the form of inner products between pairs of examples. By replacing the inner product of input data pairs with an appropriately chosen 'kernel' function, one can avoid to explicitly

establish a non-linear mapping to a high dimensional feature space. In the feature space, the size of the model is determined by the number of support vectors so that the difficulty caused by large dimensions of the input space is avoided.

Thus, linear support vector regression can now be extended to the nonlinear space, by application of the so called 'kernel trick'. In the primal weight space the function takes the form

$$f(x;\omega,b) = \omega^T\varphi(x) + b \tag{3.16}$$

The optimization problem in the primal weight space becomes

$$\begin{aligned} \min_{\omega,b,\xi,\xi^*} J_P(\omega,\xi,\xi_k^*) &= \frac{1}{2}\omega^T\omega + c\sum_{k=1}^{N}(\xi_k + \xi_k^*) \\ \textit{such that} \quad & y_k - \omega^T\varphi(x_k) - b \leq \varepsilon + \xi_k, k = 1,\ldots,N \\ & \omega^T\varphi(x_k) + b - y_k \leq \varepsilon + \xi_k^*, k = 1,\ldots,N \\ & \xi_k, \xi_k^* \geq 0, k = 1,\ldots,N. \end{aligned} \tag{3.17}$$

By replacing the inner product to a kernel function, the dual problem becomes 3.18:

$$\begin{aligned} \min_{\alpha,\alpha^*} J_D(\alpha_k,\alpha_k^*) &= -\frac{1}{2}\sum_{k,l=1}^{N}(\alpha_k - \alpha_k^*)(\alpha_l - \alpha_l^*)K(x_k,x_l) \\ & \quad -\varepsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + \sum_{k=1}^{N} y_k(\alpha_k - \alpha_k^*) \\ \textit{such that} \quad & \sum_{k=1}^{N}(\alpha_k - \alpha_k^*) = 0 \\ & \alpha_k, \alpha_k^* \in [o,c]. \end{aligned} \tag{3.18}$$

The dual representation of the target function becomes:

$$f(x) = \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)K(x,x_k) + b \tag{3.19}$$

The linear learning machine is one of the main building blocks of the system. The kernel functions are used to define the implicit feature space in which the linear learning machines operate. The danger of overfitting inherent to high dimensions is avoided by the guaranteed

generalization error bound provided by the statistical learning theory. Optimization theory gives a precise characterization of the properties of the solution which guide the implementation of efficient learning algorithms and ensure that the output hypothesis has a sparse representation. The choice of a convex cost function also results in the absence of local minima so that solutions can always be found efficiently even for training sets with hundreds of thousands of examples, while the sparse representation of the hypothesis means that the evaluation on new inputs is very fast. The size of the dual problem is independent of the dimension of the input space, but depends on the number of support vectors. So we say the primal problem is parametric while the dual problem is non-parametric. Hence, the four problems of efficiency of training, efficiency of testing, overfitting and algorithm parameter tuning are all avoided in the SVM design.

In summary, Vapnik's standard SVM possesses the properties of global optimum, sparse representation and bounded generalization risk, which can be derived directly from the solution of the optimization problem and statistical learning theory [8]. These properties make SVMs advantageous compared to Neural Networks that suffer the local optimum and over-fitting problem due to using ERM principle.

3.2.2 SVM Implementation Techniques

As we explained in the section 3.2.1, training of SVM is essentially a constrained optimization problem. The optimization problem is converted to a convex quadratic form in the dual space using Lagrange techniques. The convex property ensures that the problem has global optimum. Since the risk of local minimum is avoided, the solution can be found efficiently by the off-the-shelf optimization algorithm. In addition, the dual form indicates that the complexity of the optimization problem depends on the number of support vectors instead of the dimension of the input space, thus, this learning method can be applied in very high dimensional space.

Most of the off-the-shelf optimization algorithms employ a numerical strategy to find the minimum. The numerical strategy is starting from an arbitrary feasible point, then moving along the direction mostly improving the objective function without violating the constraints until a stopping criterion is reached. In the context of SVM formulation, the

stopping criterion can be selected from the following options [8]:

- The most straightforward criterion is to monitor the increase of the cost function. The training can be stopped when the fractional rate of the increase of the objective function falls below a given tolerance (e.g. 10^{-9}). Unfortunately, this criterion has been shown to be unreliable and in some cases can deliver poor results.

- The second way is to verify the Karush-Kuhn-Tucker conditions for the primal problem, since they are the necessary and sufficient conditions for convergence. Naturally these stopping criteria must be verified within some chosen tolerance, for example a good choice in this case is 10^{-2}.

- The third stopping criterion is to observe the gap between the primal and dual objective functions, since this difference is zero at the optimal point for convex quadratic optimization problems. This difference is called feasibility gap to distinguish it from the duality gap of an optimization problem, which is the difference between the values of the primal and dual solutions.

The conventional optimization algorithms to solve the dual problem include gradient ascent/decent, conjugate gradient, Newton method, primal dual interior-point methods. Conceptually, they are not very different in the sense that they all iteratively update the objective function to the optimum. The benefits of using these algorithms are that they are well understood and widely available in a number of commercial and freeware packages. But these traditional quadratic programming algorithms are not suitable for large size problems because of the following reasons. First, many of them require that the kernel matrix be computed and cached in memory. This requires extremely large memory. Second, the computation of kernel matrix is expensive. Third, the implementation is difficult. As the consequence, these algorithms can not be used in on-line learning. For large size problems, these approaches can be inefficient. Attempts have been made to develop methods that overcome some or all of these problems. Decomposition techniques are proposed to be used in conjunction with these optimization algorithms to reduce the space complexity [8].

As we have shown, the solution of the dual quadratic optimization problem is finally involved with support vectors corresponding to non-zero Lagrangian multipliers, i.e., to the

active constraints. For larger problems, if it is possible to find a subset of data which are an approximation of the support vectors using heuristic, it would be possible to discard all of the other data and simplify the problem. The heuristic such as decomposition starts with an arbitrary subset of data and solves the optimization problem on this subset of data. The chunk of data being optimized at a particular stage is referred to as the working set. The active set is the set of support values (α's) which is returned from the solution of the optimization problem. At each iteration, the optimization algorithm can be any of the generic quadratic program optimizer that can be directly applied to the moderate size optimization problems. The working set is iteratively updated by some criteria and the active set is gradually built until the stopping criterion is satisfied. The goal of decomposition techniques is to optimize the global problem by acting on a small subset of data at a time. The important point is to select the working set in such a way that the optimization on the working set leads to an improvement in the overall objective function. An efficient heuristic for choosing the working set at each step is to select new working set from points that contribute most of the feasibility gap or equivalently that violate the Karush-Kuhn-Tucker conditions mostly. The stopping criterion can be one of the three options explained above.

The pseudo code for the decomposition method is shown in Table 3.1:

Table 3.1: Pseudo code of the decomposition method [8]

Given training set S
$\alpha \leftarrow 0$
select an arbitrary working set $\hat{S} \subset S$
repeat
    solve optimization problem on $\hat{S}$
    select new working set from data not satisfying Karush-Kuhn-Tucker conditions
until stopping criterion satisfied
return α

Platt proposed an algorithm, called SMO [36], for solving SVM classification problems. The basic idea is to drive the decomposition technique to its extreme – utilizing a minimum working set with two data points. At each step SMO chooses two Lagrangian multipliers α_i and α_j to jointly optimize, given that all the others are fixed, and updates the α vector

accordingly. The choice of the two points is determined by a heuristic, while the optimization of the two multipliers is performed analytically. Therefore, the quadratic programming optimization algorithms are not needed. This technique addresses the aforementioned three difficulties to implement quadratic program optimizer on SVM dual problems. First, since the kernel matrix operation is not involved, the computation speed in each iteration is faster. Even though more iterations are needed to converge, the overall speed-up can be achieved. Second, the memory requirement for kernel matrix caching is not necessary. Third, the implementation of the analytical solution is much easier and no other packages of quadratic programming optimizer are needed.

Smola and Schölkopf [38, 39] extended these ideas for solving SVM regression problems.

3.2.3 LS-SVM

LS-SVM Regression

Although SVMs have many appealing properties that avoid the problems (e.g., overfitting, inefficiency of training and testing, a large set of parameters to be tuned) frequently associated with the classical supervised learning methods, they also have some drawbacks. The standard SVM requires the kernel matrix to be cached to improve the computation speed that makes online learning infeasible. Also, the fact that SVM formulation is a convex quadratic programming (QP) guarantees the global optima, but QP is still difficult to solve, especially for the learning tasks where the speed is concerned. Based on the concept and formulation of SVMs, researchers have investigated the modification and improvement of this machine for different purposes.

Least Squares Support Vector Machine (LS-SVM) is a reformulation of the standard SVM. LS-SVM models for classification and nonlinear regression are characterized by simplifying the quadratic optimization problem into a system of linear equations. Such characterizations of LS-SVM allow fast training and less storage, and thus enable its use in on-line training. LS-SVMs can be related to regularization networks and Gaussian processes. As SVMs, LS-SVMs exploit primal-dual transformations of the optimization problem. The nice properties of SVMs, such as sparseness, can be imposed to LS-SVMs where needed. A

Bayesian framework with three levels of inference has been developed to select the hyper-parameters and reduce the input space. For ultra large scale problems, on-line learning is proposed using LS-SVM [31].

LS-SVM was originally proposed by Suykens et al. [45] and aims at simplifying the formulation of the standard SVM. It is a modification of Vapnik's SVM. The nonlinear target function is the same as in 3.16 and the optimization problem can be expressed as in the formulation 3.20:

$$\begin{aligned} \min_{\omega,b,e} J_P(\omega, e) &= \frac{1}{2}\omega^T\omega + \gamma\frac{1}{2}\sum_{k=1}^{N}(e_k^2) \\ \textit{such that} \qquad & y_k = \omega^T\varphi(x_k) + b + e_k, k = 1, \ldots, N. \end{aligned} \tag{3.20}$$

In fact, this is nothing but a ridge regression cost function formulated in the feature space [8]. γ plays the same role as the regularization parameter C in SVM formulation. This LS-SVM formulation modifies Vapnik's SVM at two points. First, LS-SVM takes equality constraints instead of inequality constraints. Second, the error variable e_k was introduced in the sense of least-square minimization. These error variables play a similar role as the slack variables in SVM formulation so that relatively small errors can be tolerated [45].

In the case of linear function approximation, one could easily solve this primal problem as it involves linear equality constraints. But, when the dimension of ω becomes infinite as the dimension of the input space, the primal problem is difficult to solve. The solution is to derive the dual problem by constructing the Lagrangian for this primal problem:

$$L(\omega, b, e; \alpha) = J_P(\omega, e) - \sum_{k=1}^{N}\alpha_k(\omega^T\varphi(x_k) + b + e_k - y_k) \tag{3.21}$$

Taking the condition for optimality of the Lagrangian yields a set of linear equations shown in equation set 3.22:

$$\begin{cases} \frac{\partial L}{\partial \omega} = 0 \quad \rightarrow \quad \omega = \sum_{k=1}^{N} \alpha_k \varphi(x_k) \\ \frac{\partial L}{\partial b} = 0 \quad \rightarrow \quad \sum_{k=1}^{N} \alpha_k = 0 \\ \frac{\partial L}{\partial e_k} = 0 \quad \rightarrow \quad \alpha_k = \gamma\, e_k, \qquad k = 1, \cdots, N \\ \frac{\partial L}{\partial \alpha_k} = 0 \quad \rightarrow \quad \omega^T \varphi(x_k) + b + e_k - y_k = 0, \quad k = 1, \cdots, N. \end{cases} \tag{3.22}$$

Solving this set of linear equations in α and b, the resulting LS-SVM model for function approximation becomes as follows:

$$y(x) = \sum_{k=1}^{N} \alpha_k K(x, x_k) + b \tag{3.23}$$

As it was shown in the previous section, SVMs solve the nonlinear regression problems by means of convex quadratic programs (QP). The use of least squares and equality constraints for the models leads to solving a set of linear equations, which is easier to use than QP solvers. But, on the other hand, it has potential drawbacks such as the lack of sparseness which is indicated from the condition $\alpha_k = \gamma\, e_k$ in equation set 3.22, since the error would not be zero for most of data points. One can overcome the drawbacks using special pruning techniques for sparse approximation [45]. Pruning is a technique that can be used to improve the generalization performance by iteratively eliminating a small set (i.e. 5%) of the less relevant support vectors until the user-defined performance index degrades. If the performance becomes worse, one might check whether an additional modification of (γ, σ) (in the RBF kernel case) could improve the performance. The insight of such pruning technique comes from the pruning techniques used with NN. For MLP it is well known that by sarting from a huge network and deleting interconnection weights which are less relevant one can improve the generalization performance [4].

LS-SVMlab

The software of LS-SVM implementation in MATLAB is named LS-SVMlab and is mainly intended for use with the commercial Matlab package [31]. The present LS-SVMlab toolbox contains Matlab/C implementations for a number of LS-SVM algorithms. Functions can be

called either in a functional way or using an object oriented structure (referred to as the model), depending on the user's choice.

LS-SVMlab toolbox is one of the SVM related toolboxes we have searched and tested thoroughly. A number of functions are restricted to LS-SVMs, while the others are generally usable. Most functions can handle datasets up to 20000 data points or more. LS-SVMlab's interface for Matlab consists of a basic version and a more advanced version. The basic version is for beginners and implements the algorithms for classification, regression, time series prediction, model selection and model validation. The advanced version contains multi-class encoding techniques, a Bayesian framework, NARX models and fixed-size LS-SVM. Users can choose which version to install and use depending on their own needs.

3.3 Neural Networks

3.3.1 MLP Neural Network

Neural networks, as the name implied, are inspired by biological nervous systems. The basic idea of neural networks start from the concept of "neuron" or perceptron. According to the McCulloch-Pitts model, a neuron is modelled as a simple static nonlinear element which takes a weighted sum of incoming signals x_i and a bias term as input. The McCulloch-Pitts neuron model [45] is shown in Figure 3.2.

The output $y = h(\sum_{i=1}^{N} \omega_i x_i + b)$, where x_i is input element, $h(\cdot)$ is activation function or transfer function, w_i is the connection weight and b is a bias value. The transfer function in the figure is just an example. In fact, it can be any linear or nonlinear differentiable function. This neuron model corresponds to the biological interpretation of firing of a neuron depending on gathering and synthesizing information of incoming signals. In the perspective of machine learning, such neuron can be used to map the relationship between a pair of input and output data example.

Neural networks are composed of neurons organized in a network structure. The network structure consists of input and output layers, and several hidden layers in between. In each layer, there are a number of neurons. Between two layers, the neurons are interconnected.

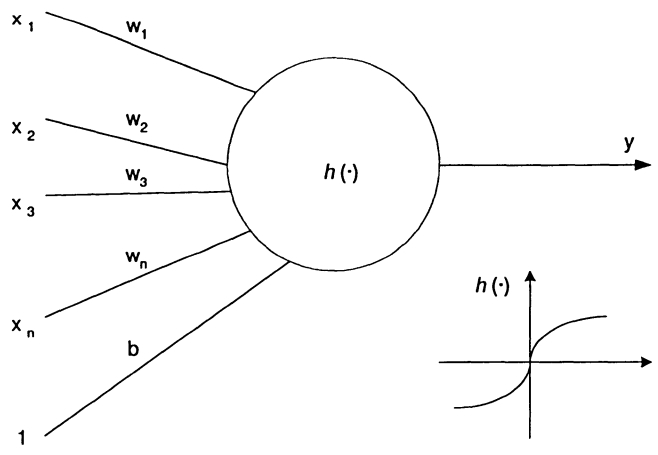


Figure 3.2: Neuron and Transfer Function [45]

By constructing such multilayer perceptron network (MLP), neural network can be very powerful. It has been mathematically proven that MLPs can approximate any continuous nonlinear function arbitrarily well over a compact interval to any degree of accuracy provided they contain one or more hidden layers [45].

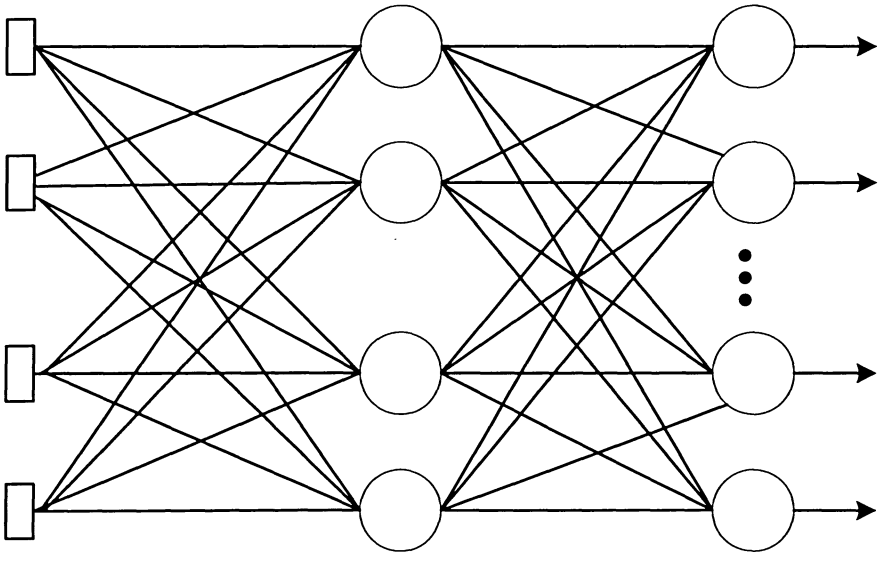


Figure 3.3: Multilayer Perceptron Network [45]

Neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Typically many such input/target pairs are used, in supervised learning,

to train a network. The function implemented by a neural network is determined by the number of layers, the number of nodes in each layer, the transfer function and the values of the connections (weights) between elements. The number of layers, the number of nodes in each layer and the transfer functions are usually selected by the designer using empirical methods. There are many algorithms that can be used to determine the weights and most of them are based on the standard optimization techniques such as conjugate gradient and Newton methods [33]. These kind of neural networks are also referred to as backpropagation neural networks (BPNN).

3.3.2 RBF Neural Network

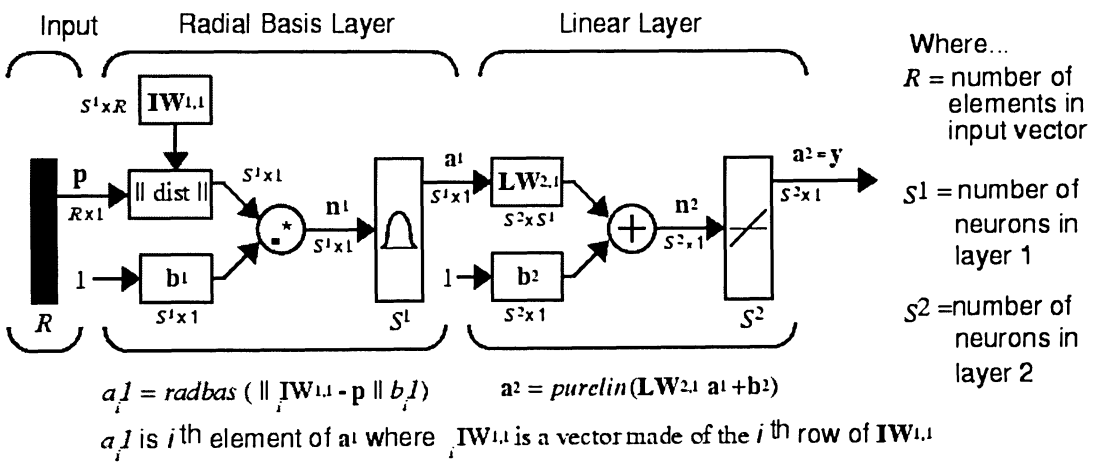


Figure 3.4: RBF Neural Network Architecture [33]

The architecture of the Radial Basis Function Neural Network (RBFNN) is shown in Figure 3.4. As shown in Figure 3.4, besides the input and output, the RBFNN consists of a hidden radial basis layer and a linear layer. The first layer, Radial Basis layer, has as many radial basis neurons as there are input vectors in the training data. The distance between an input vector P and a weight vector $IW_{1,1}$, denoted by $\|dist\|$, is passed through a transfer function in the form in Equation 3.24:

$$radbas(n) = e^{-n^2} \tag{3.24}$$

Figure 3.5 is a plot of the *radbas* transfer function.

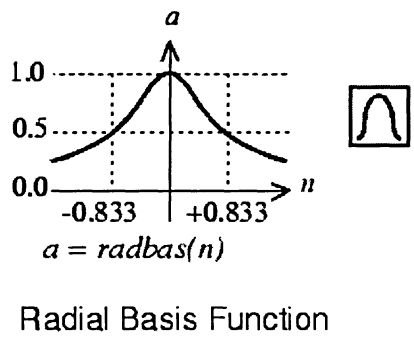


Figure 3.5: Radial Basis Function [33]

From this plot, we can see that the function *radbas* has a normal curve. The spread of the curve can be adjusted by the bias term $b1$ in Figure 3.4. The maximum 1 occurs at $n = 0$, that is, the output of radial basis layer is about 1 when the distance of the weight vector and the input vector is close to 0 (i.e, the weight vector and the input vector are identical). In contrast, radial basis neurons with weight vectors quite different from the input vector P have outputs near zero. This implies that the radial basis layer is to detect the input vectors that are close to the weight vectors.

The S_1 outputs of the radial basis layer are then sent to the second layer, linear layer, as the inputs. The second layer is simply a linear layer with a linear transfer function plotted in Figure 3.6. The input of the linear transfer function is $LW_{2,1}a_1 + b2$ as shown in Figure 3.4.

The small outputs of the first layer have only a negligible effect on the linear output neurons. Only if the output of the first layer is close to 1, its output weight in the second layer passes its value to the linear neurons in the second layer.

Thus, the final output is the weighted sum of the number of input vectors whose distance are deemed close to the radial basis layer weight vectors.

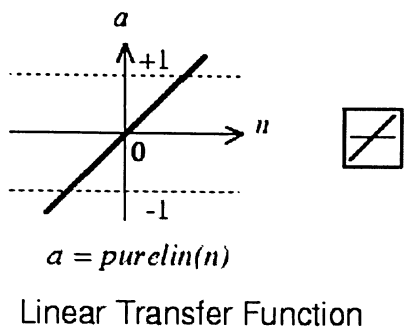


Figure 3.6: Linear Transfer Function [33]

RBFNN vs. BPNN

The RBFNN is usually designed quickly by setting the first-layer weights to the same as input vectors in the training data. Thus, the number of radial basis neurons is proportional to the size of the training data. Compared to standard feed-forward backpropagation (MLP or BPNN) networks, RBFNN may require more neurons and work best when many training data are available.

The generalization capability of BPNN lacks of enough accuracy in predictions that require extrapolation. RBFNN combining unsupervised learning like self-organized feature mapping and supervised learning like back-propagation neural network is applicable to overcome the over-fitting model problem. RBFNN is generally considered as a function approximator that can be employed in the time series forecast [5].

3.3.3 Generalized Regression Neural Network

RBFNN can be used for both classification and regression. A special neural network often used for function approximation is called Generalized Regression Neural Network (GRNN). GRNNs were introduced in 1991 by Specht [44] as a variant of RBFNN. The difference between GRNN and RBFNN is in the second layer as shown in Figure 3.7.

GRNN was introduced as a memory based neural network that would store all the training data available for a particular mapping. Suppose the GRNN is designed using input/target vector pairs, P and T. As in RBFNN, the weight vectors in the first layer are set to the

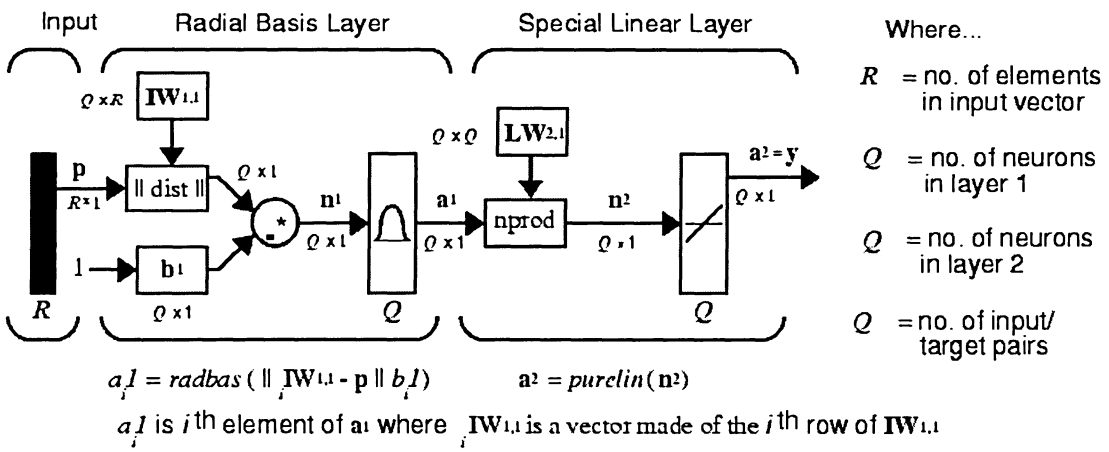


Figure 3.7: Generalized Regression Neural Network [33]

input vectors P. To be noted, in the second layer of GRNN, the weight vectors are set to the target vector T. The 'nprod' means to take the dot product. If the first layer detects a new input vector that is close to a training input vector, the second layer will output the target value of the training input vector.

The bias in the radial basis layer adjusts the smoothness of the function approximation. In short, the larger the spread is, the smoother is the approximating function.

Like many other neural architectures, GRNNs exhibit the universal approximation property for smooth functions, and can solve any approximation problem, given sufficient data. The GRNN has many advantages, but it also suffers from disadvantage.

The advantage of GRNN networks is its parallel structure and one-pass learning ability, as there is no iterative training. For this property, GRNNs can potentially be implemented in special purpose firmware, on custom-designed boards, or executed on parallel computer systems. The parallel network form could be used in applications such as learning the dynamics of a plant model for prediction or control.

However, because they must store every training sample, they are very large and require more memory. In spite of rapid training, they run relatively slowly compared with other neural architectures. Because of the high computational cost, GRNN should be used primarily for off-line applications. They also do not extrapolate, and can be used only for tasks (such as image interpolation) that can be formulated as regression problems.

GRNNs have been successfully used in approximation applications, including clinical prediction models, drug design, and ultrasound speckle modeling, but are not as commonly used as RBFNNs or backpropagation-trained networks.

GRNN vs. RBFNN

Both GRNN and RBFNN use radial basis function in the hidden layer and perform linear operations in the output layer. The main difference is that GRNN output layer performs a weighted average while the RBFNN performs a weighted sum.

Which of these two networks is better depends mainly on the application. The GRNN definition requires that it store the majority of the training data. When computational constraints are not significant, such as in stock market prediction where predictions may only be required once per day, it is probably best to use the memory based GRNN. However in most other applications, such as embedded control systems, computational efficiency is more crucial and the RBFNN would be preferred [16].

3.4 SVM vs. Neural Network

SVMs and Neural Networks are both approaches of supervised learning. Neural Networks were invented earlier and have seen a lot of successful applications in various areas, especially in the 1990's. SVMs were introduced in 1995 and have outperformed other learning methods in many applications, especially in classification problems. As the theoretical parts of SVMs and NNs were reviewed in the previous sections, in order to better understand their differences, their comparison is summarized in Table 3.2.

3.5 Prediction models

Basically, SVM is used for classification or function approximation, which involves the mapping of multi-dimensional input and output. Prediction can be treated as a special function approximation problem that maps the new output value into the previous input and output values. In this section, we introduce three prediction models.

3.5.1 Time Series Prediction model

Time series prediction is to predict one or more variables in the future point in time. From the perspective of machine learning, time series prediction is a special case of function estimation and equivalent to find the underlying functional relationship between previous values and the next value. Thus, SVMs can be used in time series prediction in the form of 3.25 :

$$\hat{y}_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k-p}) \tag{3.25}$$

The number p is referred to as embedding dimension or memory order in time series. Note that the value of p determines the dimension of inputs of the SVM model.

3.5.2 NARX model

One of the deficiencies of time series prediction is that it is unable to accommodate other meaningful input variables in system identification or dynamic modelling. An important and useful class of discrete-time nonlinear model is NARX model (Nonlinear AutoRegressive model with Exogenous Inputs) as in Equation 3.26.

$$\hat{y}_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k-p}, u_k, u_{k-1}, \ldots, u_{k-q}) \tag{3.26}$$

where u_k and y_k represent input and output of the model at time k, q and p are the input memory order and output memory order respectively, and the function f is a nonlinear

Table 3.2: Comparison between SVM and Neural Network

	SVMs	NNs
Generalization	statistical learning theory	Regularization
Risk minimization principle	SRM	ERM
Nonlinearity	Kernel function	Nonlinear transfer functions
Challenge	Choice of kernel functions	Structure of network
Parameter	fewer free parameters	more free parameters
Solution	unique and global optimum	subject to local optimum
Sparseness	Sparse solution	Not sparse
Dimensionality	Infinite dimension	Curse of dimensionality
Training time	Faster	Slower

function.

NARX model combines the power of function approximation and time series prediction. The embedded memory of the input and output variables plays an important role in learning capability and generalization performance by incorporating the historical information. The selection of input-memory and output-memory is critical for the forecasting performance. "The problem of choosing the proper memory architecture corresponds to giving a good representation of input data. A good representation can make useful information explicit and easy to extract." [28, 29]

3.5.3 NOE model

Even though the NARX model intends to model the dynamic of the system, it is inherently feedforward model since this model uses the true output values as the model input. In real prediction problem, it is usually not possible to know the true output values. For example, in the prediction of NO concentration in the nitrogen removal processes, NO is a state variable for which the actual concentration is difficult to measure. So a prediction model that can utilize the predicted value as the model input is needed. Nonlinear Output Error (NOE) model is such a recuurent model 3.27:

$$\hat{y}_{k+1} = f(\hat{y}_k,\ \hat{y}_{k-1}, \ldots, \hat{y}_{k-p},\ u_k,\ u_{k-1}, \ldots, u_{k-q}) \tag{3.27}$$

3.6 Proposed solution

As we found LS-SVM has some advantages in terms of simplicity and potential use for on-line learning, we propose to use LS-SVM as the learning machine for our target problem – predicting the NO concentration in biological nitrogen removal processes.

LS-SVM is a learning method that can solve function approximation as previously explained. In order to predict the short-term NO concentration in a short term, we need to collect the training and testing data set and apply them into appropriate prediction model. NARX and NOE models are good choices. To make it simple at this stage, we use batch

learning as the training data are all at hand. The online learning could be studied in the future work. NARX model is proposed to be used to transform the training data into an appropriate prediction form. However, during the testing, we proposed to use NOE model that made use of the predicted output values instead of the true output values because the true output values are not available in the real prediction process.

The next chapter presents the simulations that were developed for utilizing LS-SVM, NARX and NOE for the modelling of the dynamics of the NO concentration.

In order to compare the performance of our proposed methods with NN-based methods, generalized regression neural network (GRNN), was also applied to the same modeling and prediction. BPNN lacks extrapolation accuracy while RBFNN is considered more suitable for time series prediction. GRNN is a special kind of RBFNN. Our target problem is a prediction problem, so we choose GRNN to compare with LS-SVM.

Chapter 4

Simulations and Applications to Nitrogen Removal

The objective of our study is to predict the behavior of the nitrogen removal of a wastewater treatment plant with two control loops using the novel machine learning approach – LS-SVM. NARX and NOE models are utilized for describing the relationship between inputs and outputs for training and testing, respectively. In order to investigate the advantages of LS-SVM, the comparison was done with GRNN, a special Neural Network approach. The further study on adaptive control of the nitrogen removal processes can be carried on provided that accurate prediction be obtained using LS-SVM model.

4.1 Data generation

As has been shown, the essential constitute of machine learning approach is data. In order to predict the level of nitrogen removal using the machine learning method, we need the data (which reflect the behaviour of nitrogen removal processes under different conditions) to train and test the learning model. Collecting such data from a real plant is not an easy task because it takes time and requires the mobilization of a great deal of resources. Furthermore, it is impossible to obtain the data reflecting quite different influent loads during a short period of time.

An alternative to using the real measured data is to use simulated data. There are simulation software packages, such as GPS-X [10], that mostly implement the mechanistic models to simulate the operations of wastewater treatment systems. We propose to use

the simulated input and output data from GPS-X for the training and testing of LS-SVM prediction. GPS-X is a modular, multi-purpose computer program for the modeling and simulation of municipal and industrial wastewater treatment plants [10].

The central task of GPS-X is simulation. Simulations can be either steady-state or dynamic. Steady-state simulation is assuming that the system variables do not vary with time. Dynamic simulations require a solution for the set of defining differential equations which are solved numerically. GPS-X provides an easy-to-use, integrated environment for developing interactive simulations of dynamic, large-scale treatment plant models.

In GPS-X, we selected to use a built-in sample system, COST (European Cooperation in the field of Scientific and Technical Research) simulation benchmark, to generate the input and output data set. COST simulation benchmark is a comprehensive description of simulation and evaluation procedures including plant layout, simulation models and model parameters, a detailed description of disturbance to be applied during testing. A complete description of the benchmark background and how to use the simulation benchmark layout can be found in [6]. The simulation benchmark plant design comprises five reactors in series with a 10-layer secondary settling tank. The first two reactors are anoxic tanks. The following three are aerobic tanks. An internal recycle from the fifth to the first tank is designed in order to bring the nitrogen rich water back to realize nitrification and denitrification.

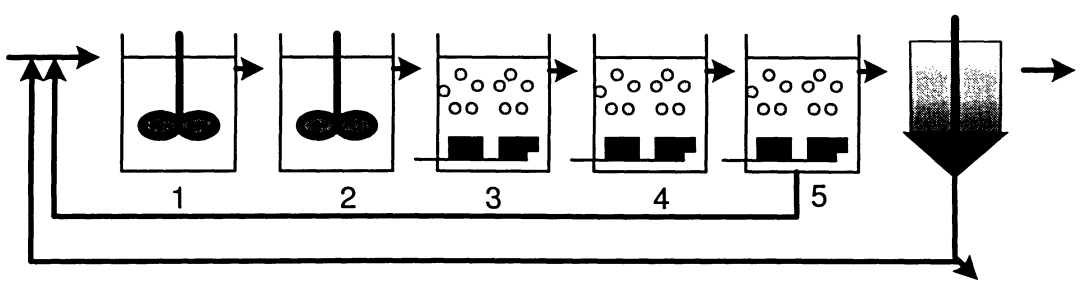


Figure 4.1: Schematic representation of '*COST simulation benchmark*' configuration showing tanks 1 & 2 mixed and unaerated, and tanks 3,4 & 5 aerated. [6]

In this process, wastewater enters the biological reactor. Here, firstly, bacterial particles are brought in contact with the organic material in the wastewater. The bacteria utilize the organics in wastewater as food. In the second stage of the reactor, bacteria known

as nitrifying bacteria predominate. These bacteria utilize ammonia as food, and convert ammonia into nitrate. After the wastewater is discharged from the aeration tank, a clarifier separates the suspended solids from the treated wastewater. Treated water is discharged into surface waters while concentrated sludge suspension is continuously withdrawn at the bottom of the secondary settler. Most of the concentrated sludge suspension is recycled and mixed again with wastewater entering the treatment plant. The excess sludge produced due to bacterial growth during degradation processes is normally discarded as a fraction of the concentrated sludge flow withdrawn at the bottom of the secondary settler and treated separately in the sludge treatment facilities of the activated sludge plant. The internal recycle from the 5th to the 1st tank is referred as nitrate internal recycle. The aim of introducing this recycle is to bring the nitrate rich water back to the first two anoxic tank and achieve denitrification there.

Two internationally accepted process models were chosen to carry out the simulations. The IAWQ's Activated Sludge Model#1 (ASM1) was chosen as the biological process model [17] in the reactor. The double-exponential settling velocity function of Takács et al.[46] was chosen as a fair representation of the settling process.

The activated sludge process aims to achieve, at minimum costs, a sufficiently low concentration of biodegradable matter in the effluent together with minimal sludge production. To do this, the process has to be controlled. The basic control strategy has two control loops, or two controllers [6]. The first loop involves controlling the dissolved oxygen (DO) level in the final tank to a setpoint of $2.0gm^{-3}$ by manipulation of the oxygen transfer coefficient (Figure 4.2). The DO sensor used in this first loop is assumed to be ideal with no delay or noise.

The second control loop involves controlling the nitrate level in the second anoxic tank to a setpoint of $1.0gm^{-3}$ by manipulaing the internal recycle flow rate (Figure 4.3). In this loop, the nitrate sensor is assumed to have a time delay of 10 minutes, with white, normally distributed (standard deviation of 0.1 gm-3), zero-mean noise. The internal recycle flow rate is constrained to a maximum of 92230 m^3d^{-1} or 1.66 times the default rate.

The built-in COST benchmark simulator in GPS-X is pre-configured and finely tuned to achieve minimum integral of the square error (ISE) [6]. The plant layout in GPS-X is

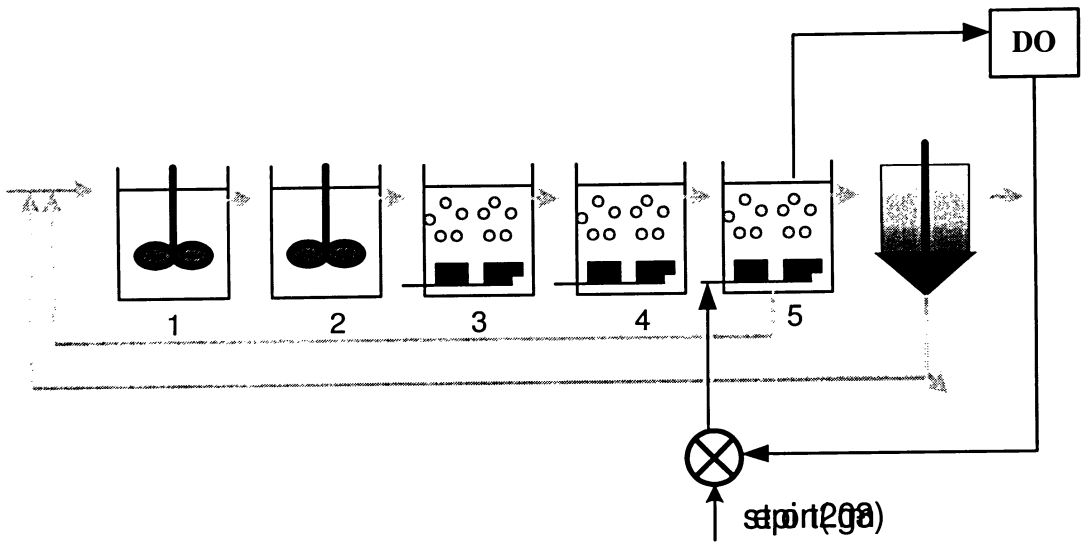


Figure 4.2: Control loop 1: DO controller [6]

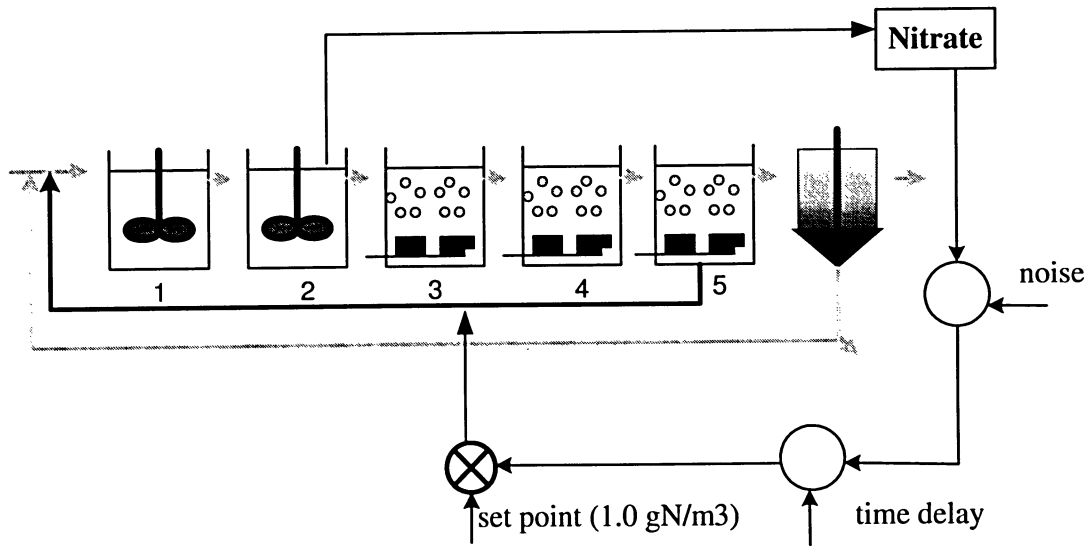


Figure 4.3: Control loop 2: Nitrate controller [6]

shown in Figure 4.4. This figure consists of four icons and their connections, which are drag-and-drop objects in GPS-X process design toolboxes. The REACTORS in this figure is only a general graphical representation of the Plug-Flow tank. The physical structure and configuration can be specified in the dialog windows associated with this Plug-Flow tank

object.

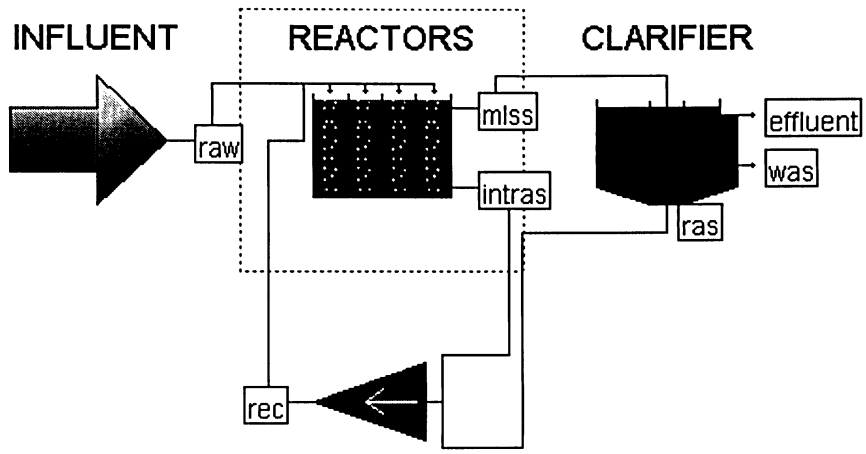


Figure 4.4: COST benchmark plant layout in GPS-X

In order to collect the training and testing data examples for the learning machine, we need to make a decision about the most appropriate input and output variables with respect to the problem under study. The variable to be predicted is known as NO concentration in the effluent of the reactor. Using domain knowledge about the wastewater treatment process, it was found that the influent flow rate, the concentration of TSS, COD, TKN and TN in the influent of the plant have significant effects on the NO concentration within the above system, so they are selected as the elements of the input vector. Because the plant is operated in closed loop with dissolved oxygen (DO) controller and nitrate controller, the variables under control are also included in the input vector. These variables are the oxygen transfer coefficient (KLa) in the final tank and the controlled nitrate value in the second anoxic tank. Thus, the input is a vector with 7 dimensions. The output is a scalar value of the concentration of NO in the effluent of the reactor (indicated by the symbol "mlss" in figure 4.4). The input and output variables as well as their description are listed in Table 4.1.

There are three influent files that have been defined in the '*simulation benchmark*' description [6, 50]. The influent files include the data of influent flow rate and influent composition.

Table 4.1: Description of the input and output variables selected for the model

Variable	Description	Symbol	Units
Q	Influent flow rate	qconraw	m^3/d
TSS	Total Suspended Solids	xraw	g/m^3
COD	Chemical Oxygen Demand	codraw	$gCOD/m^3$
TKN	Total Kjeldahl Nitrogen (organic N + NH_4^+)	tknraw	gN/m^3
TN	Total Nitrogen	tnraw	gN/m^3
KLa	Oxygen Transfer Coefficient	klalmlss(5)	$1/d$
Nitrate	Nitrate in the second anoxic tank	conno3anox2	g/m^3
NO	Nitrate and Nitrite	snomlss	gN/m^3

The data are given in the following order:

$$Time,\ S_S,\ X_{B,H},\ X_S,\ X_I,\ S_{NH},\ S_I,\ S_{ND},\ X_{ND},\ Q$$

with influent S_O, $X_{B,A}$, X_P and S_{NO} assumed to be zero. Time is given in *days*, the influent flow rate, Q, is given in m^3/day and the concentrations are given in g/m^3. The meaning of these symbols are described in Table 4.2.

Table 4.2: Description of the variables included in the influent file

Variable	Description	Units
$Time$	Elapsed time from the starting point	days
S_S	Readily biodegradable substrate	$gCOD/m^3$
$X_{B,H}$	Active heterotrophic biomass	$gCOD/m^3$
X_S	Slowly biodegradable substrate	$gCOD/m^3$
X_I	Particulate inert organic matter	$gCOD/m^3$
S_{NH}	$NH_4^+ + NH_3$ nitrogen	gN/m^3
S_I	Soluble inert organic matter	$gCOD/m^3$
S_{ND}	Soluble biodegradable organic nitrogen	gN/m^3
X_{ND}	Particulate biodegradable organic nitrogen	gN/m^3
Q	Influent flow rate	m^3/d
S_O	Dissolved oxygen	gO^2/m^3
$X_{B,A}$	Active autotrophic biomass	$gCOD/m^3$
X_P	Particulate products arising from biomass decay	$gCOD/m^3$
S_{NO}	Nitrate and Nitrite	gN/m^3

The files are representative of three disturbances: dry weather, a rain event and a storm

event. Each of the influent files contains 4 weeks (28 days) of influent data at 15-minute intervals. We select to use the last 2 weeks (14 days) of influent data to generate the required input and output data. In general, these data depict expected diurnal variations in influent flow and COD. As well, expected trends in weekly data have been incorporated. That is, much lower peak flows are depicted in the 'weekend' data, which is consistent with normal load behaviour at a municipal treatment facility [6]. The influent flow rates under three weather conditions during the last 2 weeks (14 days) are illustrated in Figure 4.5. In Figure 4.5(a), the dry weather lasts along two weeks and the influent flow rate depicts what is considered to be normal diurnal variations in flow. In Figure 4.5(b) and 4.5(c), the first week contain the same data as the dry weather data. But there is variation in the second week. Figure 4.5(b) represents a long rain event occurring in the second week. The influent flow during this rain event does not reach very high level, but the increased flow is sustained for a long period of time. Figure 4.5(c) is a variation on the dry weather with the incorporation of two storm events in the second week. The first storm event is of high intensity and short duration. The peak flow for both storms is the same, while the peak flow of the second storm is maintained over a longer period of time.

Now the question is: how can we obtain the data of the input and output variables that are of interest to us, such as TSS, COD, TKN, TN, KLa, Nitrate and NO? The answer comes from the simulation procedures in GPS-X. The system of mechanistically-based differential equations, which GPS-X automatically generates in the model building process, provides a description of the relationships among model variables. Some of these variables, referred to as state variables, are important because they define the state of the system. The state variables determine how the system behavior evolves over time. Secondary, or composite, variables are calculated from state variables and other constants, thus, always depend on how the state variables change. For example, as shown in Figure 4.6, Total Kjeldahl Nitrogen (TKN) is a composite variable calculated as the sum of several organic state variables.

State variables are the basis on which models are organized in the GPS-X libraries [10]. But composite variables are often the kind of variable typically measured in a plant.

In our problem, NO is a state variable and can be obtained as:

$$NO = S_{NO}$$

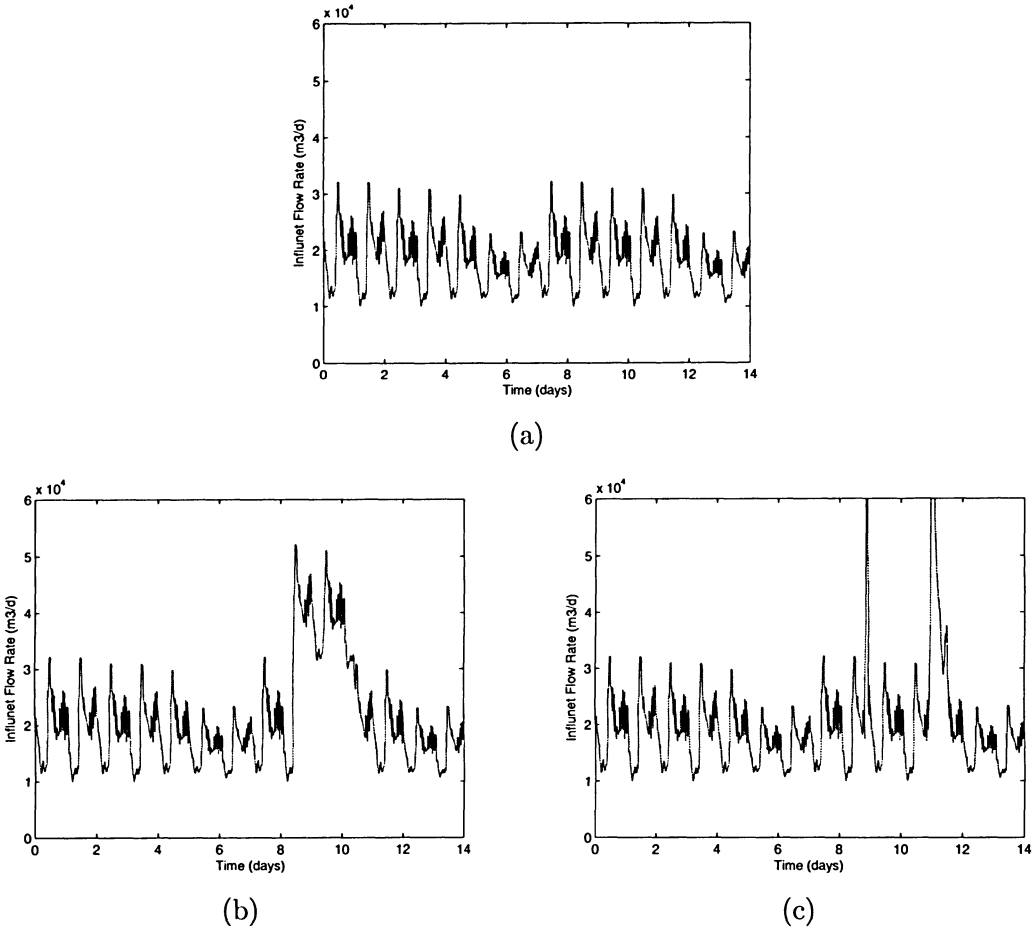


Figure 4.5: Influent flow rate under different weather conditions (a) dry (b) rain (c) storm.

The concentrations of TSS, COD, TKN and TN in the influent are composite variables in GPS-X. These composite variable calculations are defined as [6]:

$$TSS = 0.75(X_S + X_{BH} + X_{BA} + X_P + X_I)$$

$$COD = S_S + S_I + X_S + X_{BH} + X_{BA} + X_P + X_I$$

$$TKN = S_{NH} + S_{ND} + X_{ND} + i_{XB}(X_{BH} + X_{BA}) + i_{XP}(X_P + X_I)$$

$$TN = TKN + NO$$

The data about the state variables $S_S, X_{B,H}, X_S, X_I, S_{NH}, S_I, S_{ND}, X_{ND}, S_O, X_{B,A}, X_P$ and S_{NO} have been given in the influent files, as mentioned before. By running COST

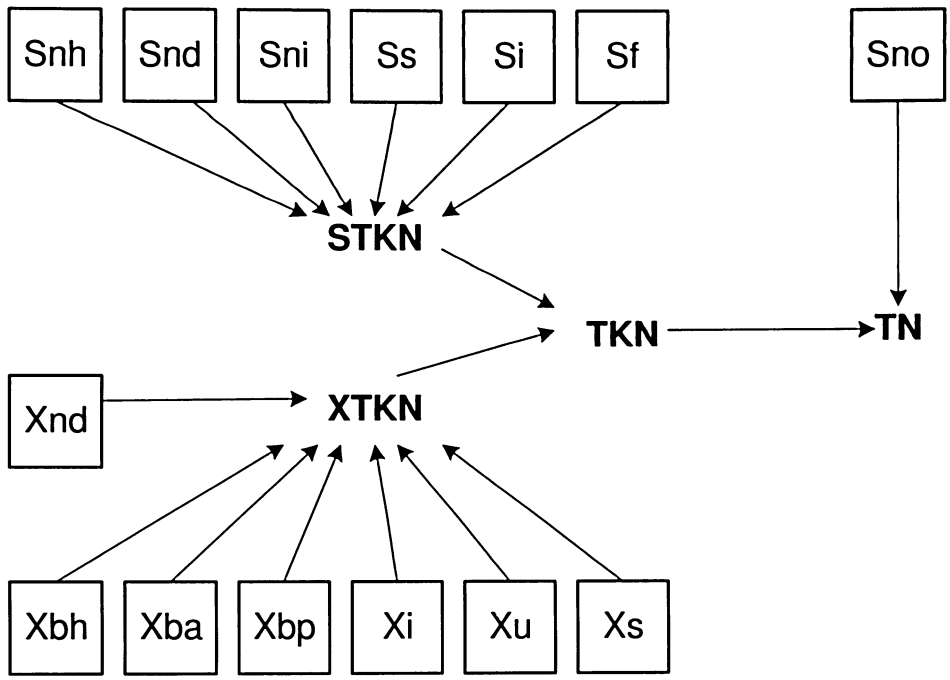


Figure 4.6: Nitrogen State (Boxed Variables) and Composite (Bold Text) Variables [10].

benchmark simulation in GPS-X using the influent files as input files, the composite variables TSS, COD, TKN, TN and NO can be calculated automatically. Since GPS-X has fully implemented COST simulation benchmark into three scenarios (dry weather, a rain event and a storm event), what we need to do is simply run the simulator under these scenarios. While the simulation is running, the data about the influent flow rate, TSS, COD, TKN and TN in the influent, KLa, Nitrogen control variables and the NO at the effluent of the reactor were saved to separate files in order to be used later as input and output for LS-SVM model. A snapshot of the interface of simulation is shown in Figure 4.7. The windows in the left display the variations of the influent flow and its composition, as well as the controller status. The windows on the right-top corner show the response of the specified variables in the effluent. The windows on the right-bottom corner plot the dynamic variations of the variables of interest, such as influent flow rate, TSS, COD, TKN, and NO.

The recorded concentrations of NO in the effluent under three weather conditions are depicted in Figure 4.8, which indicates that the concentrations of NO also have diurnal pattern and weekly trend. Note that the concentration of NO under rain or storm event is

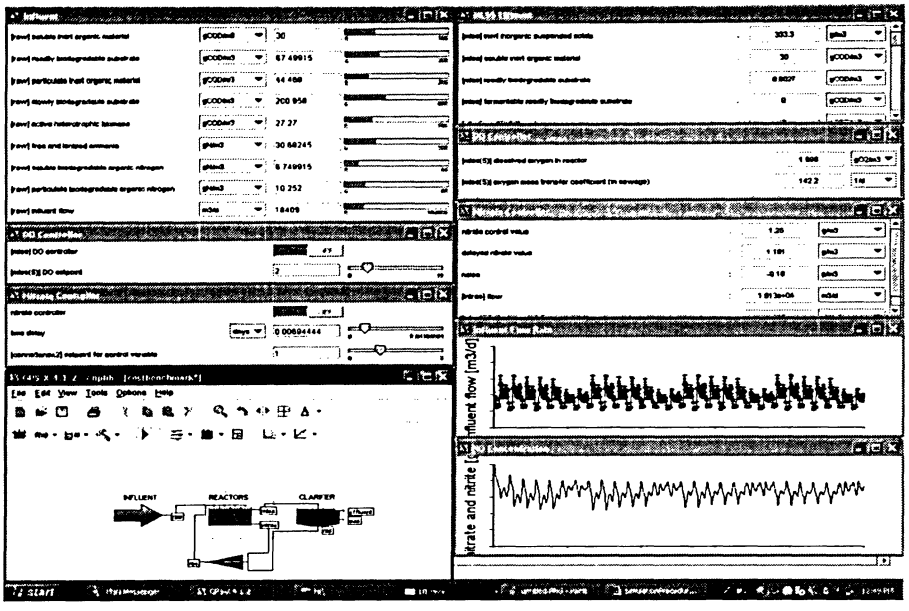


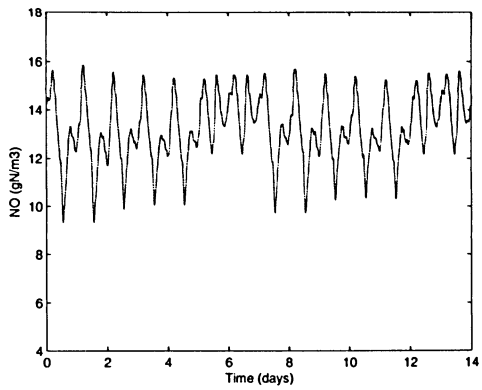
Figure 4.7: COST benchmark Simulation Interface in GPS-X

reduced because the wastewater is diluted by the excess water flow due to the rain or storm.

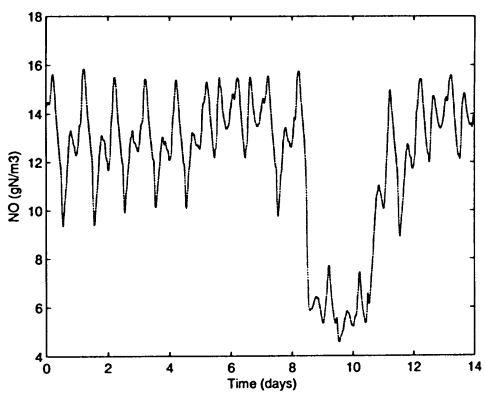
4.2 Simulation procedures

As discussed in section 3.2.3, in terms of simplicity, memory requirement and potential for implementing in on-line learning, LS-SVM has more advantages than standard SVM. Thus, we proposed to use LS-SVM combined with the NARX model to predict the target variable. Now that we have obtained the simulated input and output data from GPS-X, we can use these data as the training and testing data for LS-SVM prediction. The NARX model was employed to transform the input and output into a more suitable state space in order to accommodate the historical data and extract the information effectively. LS-SVM MATLAB Toolbox was then used to train the LS-SVM learning machine and test the prediction [45]. The simulation approach includes five steps and the procedure is illustrated in Figure 4.9.

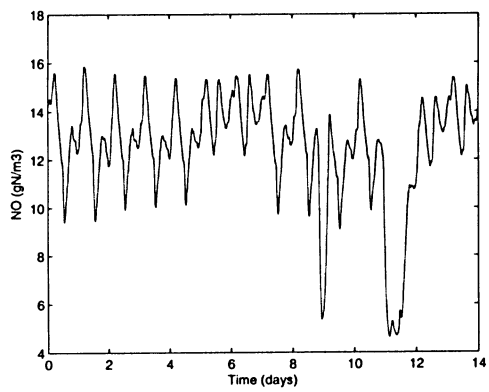
We have two objectives by doing extensive simulations following the first four steps in Figure 4.9. The first objective is to compare the prediction of the NO concentration under different weather conditions using LS-SVM. The second objective is to investigate how methods in neural network flavor can predict the NO concentration under the same



(a)



(b)



(c)

Figure 4.8: NO concentration in ML under different weather conditions (a) dry (b) rain (c) storm.

settings as LS-SVM. We chose GRNN as the counterpart in neural network to LS-SVM, because GRNN is often used for function approximation.

In order to do the simulations associated with these objectives, we need to determine the following settings for both LS-SVM and GRNN:

- Training data size
- Memory order of input and output

For training LS-SVM, the following parameters should be determined:

- Regularization parameter (i.e., γ in Equation (3.20))

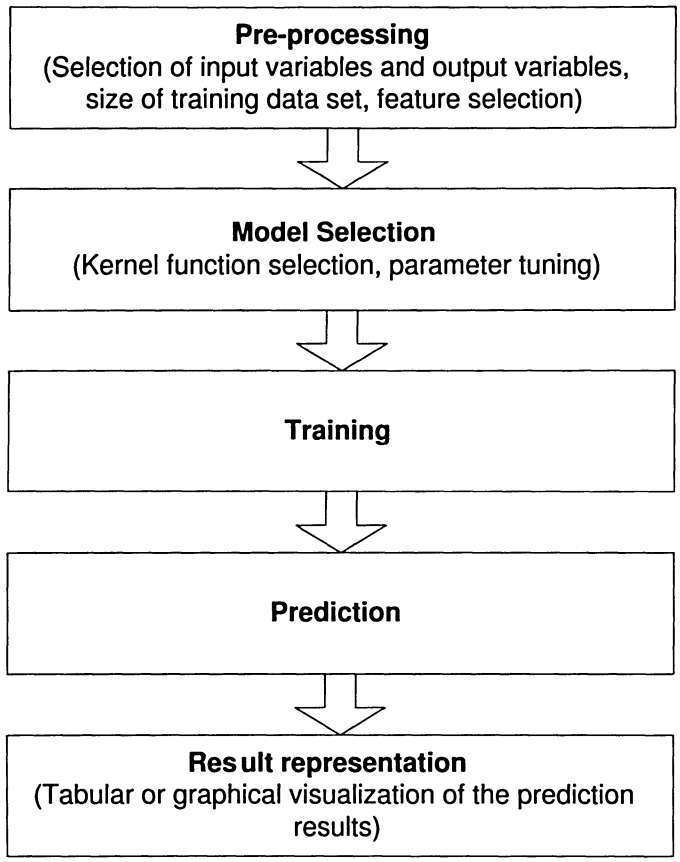


Figure 4.9: Simulation approach of prediction using LS-SVM

- Kernel function and kernel parameter

For training GRNN, one parameter needs to be determined:

- Spread parameter

The details of each step will be explained in the following subsections. To compare the performance of LS-SVM and GRNN, the discussion will emphasize on the differences between the simulation procedures and the results.

4.2.1 Pre-processing

Pre-processing aims to find the appropriate input and output presentation, determine the appropriate size of training and testing data, scaling or normalizing data. In our simulations,

transforming the input and output data and determining the training and testing data size are identical for both LS-SVM and NN.

Selection of the training data size

Selecting the training data size is not a simple issue and there isn't any method that proves to be effective. The general opinion is more training data leads to more accurate result. The size of the whole data set we have collected is $2 \times 24 \times 7 \times 4 = 1344$, since 2 weeks data were collected at a 15-minute interval. It is not a large data set. As our goal is to predict the future NO concentration and examine the generalization capacity of LS-SVM and GRNN, we decided to use the first week's data under dry weather as the training data and the second week's data as the testing data. That means we will use LS-SVM and GRNN trained by one week's data under dry weather to predict the NO concentration in the effluent data under dry weather, rain event and storm event. This seems beyond the ability of the empirical models that are thought their uses are limited to the data ranges over which they are fitted.

NARX transformation

In section 4.1, we explained how the input and output variables are selected. It is from the domain knowledge that we choose 7 variables as input and 1 variable as output. The input variables are Influent Flow Rate, TSS, COD, TKN, TN, KLa and Nitrate control variable. These variables form an input vector, while the output NO is a scalar variable. Our objective is to predict the NO concentration based on the previous input and output data. Because the prediction is highly correlated to the historical data, we proposed to use NARX model to transform the input and output variables into a more appropriate space to involve the previous values. The NARX transformation requires the selection of input and output memory orders. Since there isn't any general method that can effectively be used to select the memory order, we decided to use a trial and error approach to determine the memory order of the input and output variables for the NARX transformation.

After the proper memory order is determined, the NARX transformation can be implemented by manipulating matrices in MATLAB using a function called windowizeNARX. The function syntax is:

`[Xw,Yw,xdim,ydim,n]=windowizeNARX(X,Y,xdelays,ydelays,steps);`

Outputs

`Xw`	Matrix of the data used for input including the delays
`Yw`	Matrix of the data used for output including the next steps
`xdim(*)`	Number of dimensions in new input
`ydim(*)`	Number of dimensions in new output
`n(*)`	Number of new data points

Inputs

`X`	$N \times m$ vector with input data points
`Y`	$N \times d$ vector with output data points
`xdelays`	Number of lags of X in new input
`ydelays`	Number of lags of Y in new output
`steps(*)`	Number of future steps of Y in new output (by default 1)

where the variables with (*) are options that are not necessary to be specified. N is the number of the original training data. m is the dimension of the original input data and d is the dimension of the original output data.

For example, given 5 values of the input vectors $[Q\ TSS\ COD\ TKN\ TN\ Kla\ Nitrate\ ]$ and the corresponding 5 values of the output variable NO, we get

$$\mathtt{X} = \left[\begin{array}{c} \overbrace{\begin{array}{ccccccc} Q1 & TSS1 & COD1 & TKN1 & TN1 & Kla1 & Nitrate1 \\ Q2 & TSS2 & COD2 & TKN2 & TN2 & Kla2 & Nitrate2 \\ Q3 & TSS3 & COD3 & TKN3 & TN3 & Kla3 & Nitrate3 \\ Q4 & TSS4 & COD4 & TKN4 & TN4 & Kla4 & Nitrate4 \\ Q5 & TSS5 & COD5 & TKN5 & TN5 & Kla5 & Nitrate5 \end{array}}^{m=7} \end{array}\right] \Bigg\} N = 5$$

$$\mathtt{Y} = \left[\begin{array}{c} \overbrace{NO1}^{d=1} \\ NO2 \\ NO3 \\ NO4 \\ NO5 \end{array}\right] \Bigg\} \mathtt{N} = 5$$

If we transform X and Y with input memory order 1 and output memory order 1, the function `windowizeNARX` can be called using the following syntax:

```
[Xw Yw, xdim, ydim, n]=windowizeNARX(X, Y, 1, 1)
```

The transformation result is:

$$\mathtt{Xw} = \left[\begin{array}{ccc} \overbrace{Q1\,TSS1\,\ldots\,Nitrate1}^{m=7} & \overbrace{Q2\,TSS2\,\ldots\,Nitrate2}^{m=7} & \overbrace{NO1;}^{d=1} \\ Q2\,TSS2\,\ldots\,Nitrate2 & Q3\,TSS3\,\ldots\,Nitrate3 & NO2; \\ Q3\,TSS3\,\ldots\,Nitrate3 & Q4\,TSS4\,\ldots\,Nitrate4 & NO3; \\ Q4\,TSS4\,\ldots\,Nitrate4 & Q5\,TSS5\,\ldots\,Nitrate5 & NO4; \end{array}\right] \Bigg\} n = 4$$

$$\mathtt{Yw} = \left[\begin{array}{c} NO2 \\ NO3 \\ NO4 \\ NO5 \end{array}\right] \Bigg\} n = 4$$

`xdim`$= 7 + 7 + 1 = 15$, `ydim`$= 1$, `n`$=4$.

The resulting matrices Xw and Yw will be used as the new input and output vectors appropriate for the training of LS-SVM and GRNN. This transformation should be applied to both training and testing data.

NOE prediction

The training input and output data are resulted from the NARX transformation. For the testing data, NARX and NOE models should be used in combination by employing function windowizeNARX as well. Firstly, the testing data should be transformed by NARX model as in the previous section. Secondly, when test the prediction, the predicted output value should be iteratively integrated into the NARX transformed input vector to predict the next output value using NOE model. For example, in the testing data set, given 5 values of the input vectors [Q TSS COD TKN TN Kla $Nitrate$] and the corresponding 5 values of the output variable NO, we get

$$\mathtt{X} = \left[\begin{array}{cccccc} \overbrace{Q6 & TSS6 & COD6 & TKN6 & TN6 & Kla6 & Nitrate6}^{m=7} \\ Q7 & TSS7 & COD7 & TKN7 & TN7 & Kla7 & Nitrate7 \\ Q8 & TSS8 & COD8 & TKN8 & TN8 & Kla8 & Nitrate8 \\ Q9 & TSS9 & COD9 & TKN9 & TN9 & Kla9 & Nitrate9 \\ Q10 & TSS10 & COD10 & TKN10 & TN10 & Kla10 & Nitrate10 \end{array}\right] \Bigg\} N = 5$$

$$\mathtt{Y} = \left.\begin{bmatrix} \overbrace{NO6}^{d=1} \\ NO7 \\ NO8 \\ NO9 \\ NO10 \end{bmatrix}\right\} \mathtt{N} = 5$$

We transform X and Y with the same input memory order 1 and output memory order 1 as the transformation for the training data using NARX transformation, the function *windowizeNARX* can be called using the following syntax:

```
[Xt Yt, xdim, ydim, n]=windowizeNARX(X, Y, 1, 1)
```

The transformation result is:

$$\mathtt{Xt} = \left.\begin{bmatrix} \overbrace{Q6\,TSS6\,\ldots\,Nitrate6}^{m=7} & \overbrace{Q7\,TSS7\,\ldots\,Nitrate7}^{m=7} & \overbrace{NO6;}^{d=1} \\ Q7\,TSS7\,\ldots\,Nitrate7 & Q8\,TSS8\,\ldots\,Nitrate8 & NO7; \\ Q8\,TSS8\,\ldots\,Nitrate8 & Q9\,TSS9\,\ldots\,Nitrate9 & NO8; \\ Q9\,TSS9\,\ldots\,Nitrate9 & Q10\,TSS10\,\ldots\,Nitrate10 & NO9; \end{bmatrix}\right\} n = 4$$

$$\mathtt{Yt} = \left.\begin{bmatrix} NO7 \\ NO8 \\ NO9 \\ NO10 \end{bmatrix}\right\} n = 4$$

If we use the first transformed input vector

$$[Q6\;TSS6\;\ldots\;Nitrate6\;\;Q7\;TSS7\;\ldots\;Nitrate7\;\;NO6]$$

as the starting values to predict the next value of output $N\hat{O}7$, then $N\hat{O}7$ should replace $NO7$ in the transformed input vector

$$[Q7\;TSS7\;\ldots\;Nitrate7\;\;Q8\;TSS8\;\ldots\;Nitrate8\;\;NO7]$$

and the resulted new testing input vector

$$[Q7\;TSS7\;\ldots\;Nitrate7\;\;Q8\;TSS8\;\ldots\;Nitrate8\;\;N\hat{O}7]$$

is used to predict the next value of output $N\hat{O}8$, and so on.

Finnaly, the predicted output vector

$$\hat{\mathtt{Yt}} = \left.\begin{bmatrix} N\hat{O}7 \\ N\hat{O}8 \\ N\hat{O}9 \\ N\hat{O}10 \end{bmatrix}\right\} n = 4$$

can be compared to the desired output vector Yt and the error can be calculated.

Data scaling

Before training, it is often useful to scale the inputs and targets so that they always fall within a specified range. In LS-SVM MATLAB toolbox, certain preprocessing steps such as scaling or normalizing data can be done automatically by specifying the '*preprocess*' option in the commands used for training.

In implementing GRNN in MATLAB, scaling network inputs and targets should be done by using appropriate functions. In our simulations, in order to normalize the training set, function `prestd` is used in the syntax:

```
[pn,meanp,stdp,tn,meant,stdt] = prestd(p,t);
```

The original network inputs and targets are given in the matrices p and t. The normalized inputs and targets, `pn` and `tn` that are returned will have zero mean and unity standard deviation. The vectors `meanp` and `stdp` contains the mean and standard deviations of the original inputs, and the vectors meant and `stdt` contains the mean and standard deviation of the original data. After the network has been trained, these vectors should be used to transform any future inputs that are applied to the network.

If `prestd` is used to scale both the inputs and targets, then the output of the network is trained to produce outputs with zero mean and unity standard deviation. To convert these outputs back into the same units that were used for the original targets, the routine `poststd` should be used:

```
anew = poststd(anewn,meant,stdt);
```

Dimension Reduction

In some situations, the dimension of the input vector is large, but the components of the vectors are highly correlated (redundant). It is useful in this situation to reduce the dimension of the input vectors.

In LS-SVM, Automatic Relevant Dimension (ARD) technique can be used to reduce the dimension of the input vectors. However, this step is not used in the LS-SVM simulations due

to some difficulties to run the relative functions in LS-SVMlab toolbox. We will investigate this problem in the future work.

In GRNN, an effective procedure for performing this operation of dimension reduction is principal component analysis (PCA). This technique has three effects: it orghogonalizes the components of the input vectors (so that they are uncorrelated with each other); it orders the resulting orthogonal components (principal components) so that those with the largest variation come first; and it eliminates those components that contribute the least to the variation in the data set. The following statement illustrates the use of `prepca`, which performs the principal component analysis:

```
[ptrans,transMat] = prepca(pn,0.02);
```

Note that `pn` is the normalized input vectors using `prestd`, so that they have zero mean and unity variance. Normalization is a standard procedure when using principal components. In this example, `0.02` means that `prepca` eliminates those principal components that contribute less than 2% to the total variation in the data set. In our simulations, the dimension of the input vector for training is 15 as shown in NARX transformation in 4.2.1. Performing PCA on the input vector with respect to the parameter 0.02 resulted in an input vector with dimension reduced from 15 to 4. The matrix `transMat` contains the principal component transformation matrix. After the network has been trained, this matrix should be used to transform any future inputs that are applied to the network.

`meanp`, `stdp`, `meant`, `stdt`, and `transMat` effectively become a part of the network, just like the network weights and biases. The normalized input vectors `pn` multiplied by the transformation matrix `transMat` become the transformed input vectors `ptrans` to be used to train the network.

4.2.2 Model selection

LS-SVM model selection

Since we decided to use LS-SVM model, the model selection involves choosing kernel function and turning the kernel parameter and regularization parameter.

Among the commonly used kernel functions listed in equations 3.12–3.15, RBF kernel is a better choice. The linear kernel is too simple to use and can't approximate the nonlinear case. Furthermore, the linear kernel is a special case of RBF [26]. The sigmoid kernel behaves like RBF for certain parameters [30], but sigmoid kernel does not satisfy the Mercer condition under some parameters [48]. The polynomial kernel has more hyperparameters than the RBF kernel. In addition, the RBF kernel has less numerical difficulties [19].

According to Baesens [1], RBF kernel has the best classification performance compared to linear and polynomial kernels. In our function approximation problem, we also selected Radio Basis Function (RBF) kernel because RBF kernel behaves well when there is little information about the training data set.

In this case, the parameters to be tuned are regularization parameter γ in LS-SVM formulaion (3.20) and kernel parameter σ^2 in RBF kernel (3.14). In LS-SVM MATLAB toolbox, given the training data, the hyperparameters (γ, σ^2) can be determined on a validation set using the following code:

```
[gam,sig2] = tunelssvm(Xi,Yu,'function',10,50,'RBF_kernel',[ ],
             'gridsearch',,'validate',Xtra,Ytra, Xval, Yval);
```

In the function `tunelssvm`, `Xi`, `Yu` are the transformed input and output vector used for training. `'function'` means it is a function approximation problem. 10 and 50 are the initial values selected for the regularization parameter γ and RBF kernel parameter σ^2. `'gridsearch'` means the method to search for the jointly optimal parameters `gam` and `sig2`. `'validate'` means using validation to evaluate the two parameters `gam` and `sig2`. The tuning process is to select the optimal values of (γ, σ^2) within a specified range. The training data are divided into two distinct set: {`Xtra, Ytra`} for training and {`Xval, Yval`} for validation. The parameter (γ, σ^2) selected using a certain of search method is applied to the training data set ({`Xtra, Ytra`}) , then the prediction error on the validation set ({`Xval, Yval`}) is calculated. By comparing the prediction errors resulted from each parameter set (γ, σ^2), the parameter set leading to the minimum prediction error is regarded as the optimal.

Neural Network model selection

In using generalized regression neural network (GRNN), there is only one parameter `spread`, which determines the width of an area in theinput space to which each neuron responds. In our simulations of prediction using GRNN, `spread` is set to default value 1.

4.2.3 Training and Prediction

In the preprocessing step, the original data were transformed to a new set of input and output pairs using NARX model. These new input and output data pairs can be fed to the training function to return a model that can map the relationships between the transformed input and output.

Our target prediction problem can be modeled as a NOE prediction problem. NOE prediction can be considered as a regression. For regression, the output is the system response to the input. For prediction, the output is the future value of the target variable, which corresponds to the previous values of the input and output variables. Thus, within the implementation of NOE prediction, the regression algorithm is actually used.

LS-SVM Training and Prediction

Using the NARX transformed data in preprocessing step and the results of model selection step, the training of LS-SVM for function approximation (regression) can be realized using the following code:

```
model = trainlssvm(X,Y,'function estimation', gam,sig2,
                   'RBF_kernel','preprocess');
```

X and Y are matrices holding the training input and training output. In our case, they are NARX transformed input and output. 'function estimation' indicates this is a function approximation model. 'RBF_kernel' specifies the kernel type. `gam` is the regularization parameter. The effect of adjusting `gam` to lower value emphasizes on minimizing of the complexity of the model, while higher `gam` value stresses good fitting of the training data points. `sig2` is the parameter of the RBF kernel. A large `sig2` indicates a stronger smoothing. 'preprocess'

means the preprocessing of the training data is performed automatically. `model` is the object oriented representation of the LS-SVM model.

After the LS-SVM model is trained, iterative prediction can be done using NOE prediction model. Every predicted output value will be returned to be used together with the transformed input for predicting the next output value. Since the prediction function in LS-SVM MATLAB Toolbox can only perform time series prediction one step ahead, we modified it into an iterative prediction function called `LSSVM_predict_NARX` and it is described as follows:

```
Yp = SVM_predict_NARX(model, Xt, Xf, nb);
```

Outputs

`Yp`	`nb` × 1 matrix with the predictions

Inputs

`model`	Object oriented representation of the LS-SVM model
`Xt`	matrix of the starting points for the prediction
`Xf`	matrix of the transformed inputs for the prediction
`nb`	Number of outputs to predict

The predicted output `Yp` is evaluated on the input points contained in `Xf` by default using a function `simlssvm`. The syntax of `simlssvm` is:

```
Yt = simlssvm (model, Xt)
```

The matrix `Xt` represents the points for which one wants to make a prediction. `model` is the object oriented representation of the LS-SVM model returned from training. `simlssvm` is iteratively called within `SVM_predict_NARX` to evaluate the input points that integrate the previously predicted output value as we described in section 4.2.1.

GRNN Training and Prediction

The normalized and transformed input and output data, denoted by `ptrans` and `tn`, are used to train a generalized regression RBF network (GRNN) using function newgrnn:

```
net = newgrnn(ptrans,tn);
```

net represents the trained GRNN and is used to predict the future value of the target output (NO concentration). The iterative prediction function called NN_GRNN_predict_NARX with the same flavour as SVM_predict_NARX is implemented. Besides, the normalization vectors Meanp, stdp, meant,stdt, and the principal component transformation matrix transMat resulted from the preprocessing step are transferred as arguments to make corresponding manipulations on the testing input and convert the output back to the original unit. The syntax of NN_GRNN_predict_NARX is:

```
prediction = NN_GRNN_predict_NARX(net,Xt,Xf,nb,
                                  meanp,stdp,meant,stdt,transMat);
```

The returned prediction is a vector containing the specific number (nb) of predicted output values. Each predicted value is evaluated on the input points contained in Xf by default using a function sim. The syntax of sim is:

```
anewn = sim(net, pnewntrans)
```

where anewn is the output value evaluated at input value pnewntrans and needs to be converted back to the original unit by:

```
anew = poststd(anewn,meant,stdt)
```

4.2.4 Results visualization

The performance of the prediction is estimated by Mean Square Error (MSE), for LS-SVM and GRNN respectively. The result visualization aims at analyzing the simulation results associated with different objectives and illustrate the outcomes of the analysis using tables or plots in order to make the simulation results easier to understand and fulfill the research objectives. The visualized results of the simulations are illustrated in the following sections.

4.3 Simulation results

4.3.1 LS-SVM prediction

Memory order selection

The emphasis of this simulation is on exploring the effects of input and output memory order of NARX model on the performance of LS-SVM prediction under different weather conditions. 672 training data examples over the first week (representing dry weather) were used to train the LS-SVM with RBF kernel and predict the next 672 values of NO concentration over the second week for three different weather conditions (representing dry weather, rain event and storm event respectively). Mean Square Error (MSE) was used to measure the prediction accuracy. MSE is calculated as in Equation 4.1:

$$MSE = \frac{\sum_{i=1}^{L} e_i^2}{N} \tag{4.1}$$

where L is the number of testing data points, e_i is the difference between the predicted value and the desired value.

In Table 4.3, the prediction errors with different combinations of input and output memory orders are listed for three weather conditions. These results indicate that LS-SVM using input memory order 3 and output memory order 1 has the best performance for predicting NO concentration under the dry weather. Under rain or storm event, the optimal input and output memory order are both 1.

LS-SVM prediction results

Figure 4.10 shows the comparison of predicted and actual NO concentrations using the optimal input and output memory order selected in the previous subsection under three weather conditions. The solid line represents the actual concentration and the dashed line the predicted value.

Table 4.3: LSSVM prediction error of NO concentration (gNm^{-3}) under different weather conditions (a) dry (b) rain (c) storm.

Input memory order→ Output memory order↓	0	1	2	3
0	0.5204	0.3624	0.3104	0.2827
1	0.2793	0.1550	0.1208	0.1049
2	0.3338	0.1924	0.1423	0.1112
3	0.2473	0.1693	0.1389	0.1166

(a)

Input memory order→ Output memory order↓	0	1	2	3
0	8.0022	7.2378	6.5409	6.1016
1	4.6539	1.7146	2.5735	3.3245
2	6.0985	2.2491	2.0145	2.3592
3	6.0974	2.7482	2.4547	2.4989

(b)

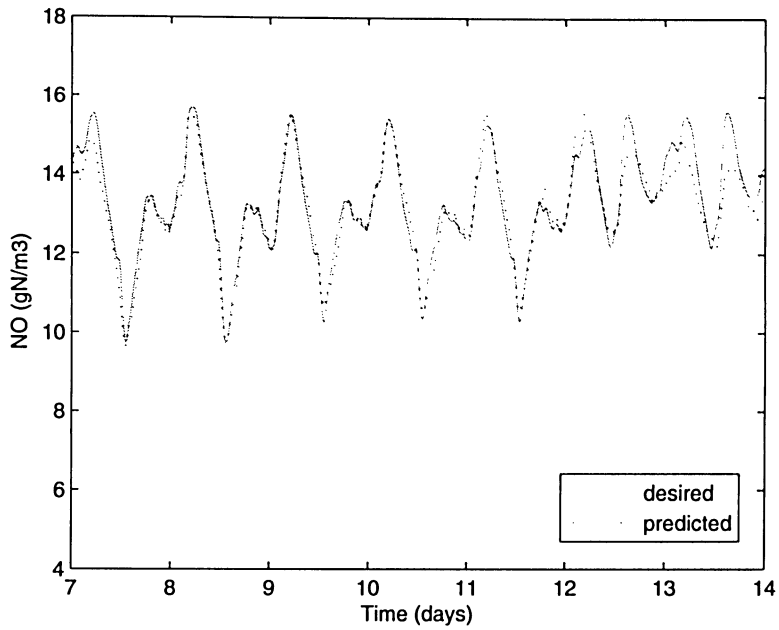
Input memory order→ Output memory order↓	0	1	2	3
0	5.0235	4.3416	4.0412	3.9068
1	2.6405	2.0437	2.6062	3.1143
2	3.532	2.5308	2.6585	3.0345
3	3.7025	2.8422	2.9338	3.1949

(c)

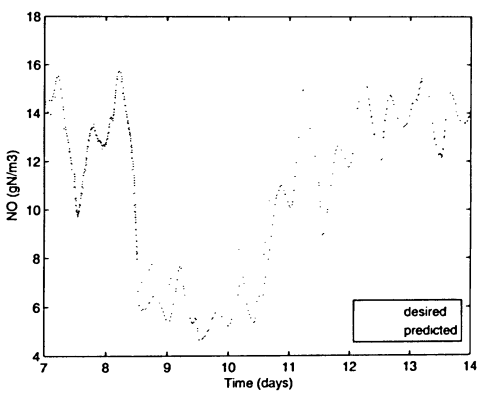
4.3.2 GRNN prediction

Memory order selection

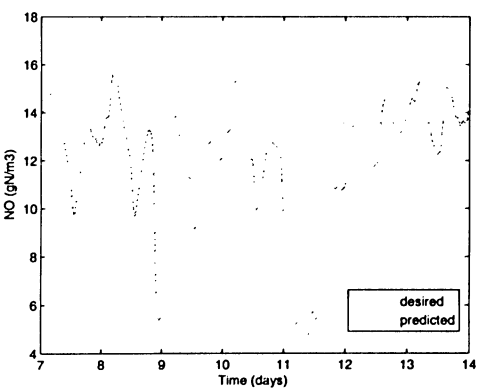
The objective of this simulation is to find the optimal input and output memory order of NARX model for GRNN prediction under different weather conditions. The simulation settings and procedures are exactly the same as those of LS-SVM prediction. 672 training data examples over the first week (representing dry weather) was used to train the GRNN and predict the next 672 values of NO concentration over the second week for three different



(a)



(b)



(c)

Figure 4.10: The comparison of predicted and desired NO concentration in ML under different weather conditions using LS-SVM (a) dry (input memory order:3, output memory order:1) (b) rain (input memory order:1, output memory order:1) (c) storm (input memory order:1, output memory order:1).

weather conditions (dry weather, rain event and storm event). Mean Square Error (MSE) was used to measure the prediction accuracy. In Table 4.4, the prediction errors with different combinations of input and output memory orders are listed for three weather conditions.

These results indicate two aspects that are different from LS-SVM. First, GRNN obtains comparable prediction accuracy under dry whether but can't predict NO concentrations under the rain and storm events very well. Second, increasing the input and output memory order doesn't seem to significantly improve the prediction performance under any of the three weather conditions, even though a little bit smaller prediction errors occur at larger input and output memory orders.

Table 4.4: GRNN prediction error of NO concentration (gNm^{-3}) under different weather condition (a) dry (b) rain (c) storm.

Input memory order→ Output memory order↓	**0**	**1**	**2**	**3**
0	0.4697	0.3727	0.3384	0.3067
1	0.4268	0.3453	0.3139	0.2866
2	0.4135	0.3303	0.2981	0.2811
3	0.3957	0.3132	0.2811	0.2662

(a)

Input memory order→ Output memory order↓	**0**	**1**	**2**	**3**
0	17.1464	16.1249	17.0145	17.1079
1	15.9773	14.7584	14.2671	13.8830
2	15.8655	14.9118	14.3167	13.8022
3	15.6117	14.9126	14.4225	13.8353

(b)

Input memory order→ Output memory order↓	**0**	**1**	**2**	**3**
0	7.5397	7.1705	7.6782	6.5200
1	7.2219	6.7170	6.4900	6.3764
2	7.1232	6.5102	6.2255	6.1887
3	6.8724	6.4441	6.1766	6.1610

(c)

GRNN prediction results

Figure 4.11 shows the comparison of predicted and actual NO concentrations using the same input and output memory order as those used with LS-SVM under three weather conditions. The solid line represents the actual concentration and the dashed line the predicted value.

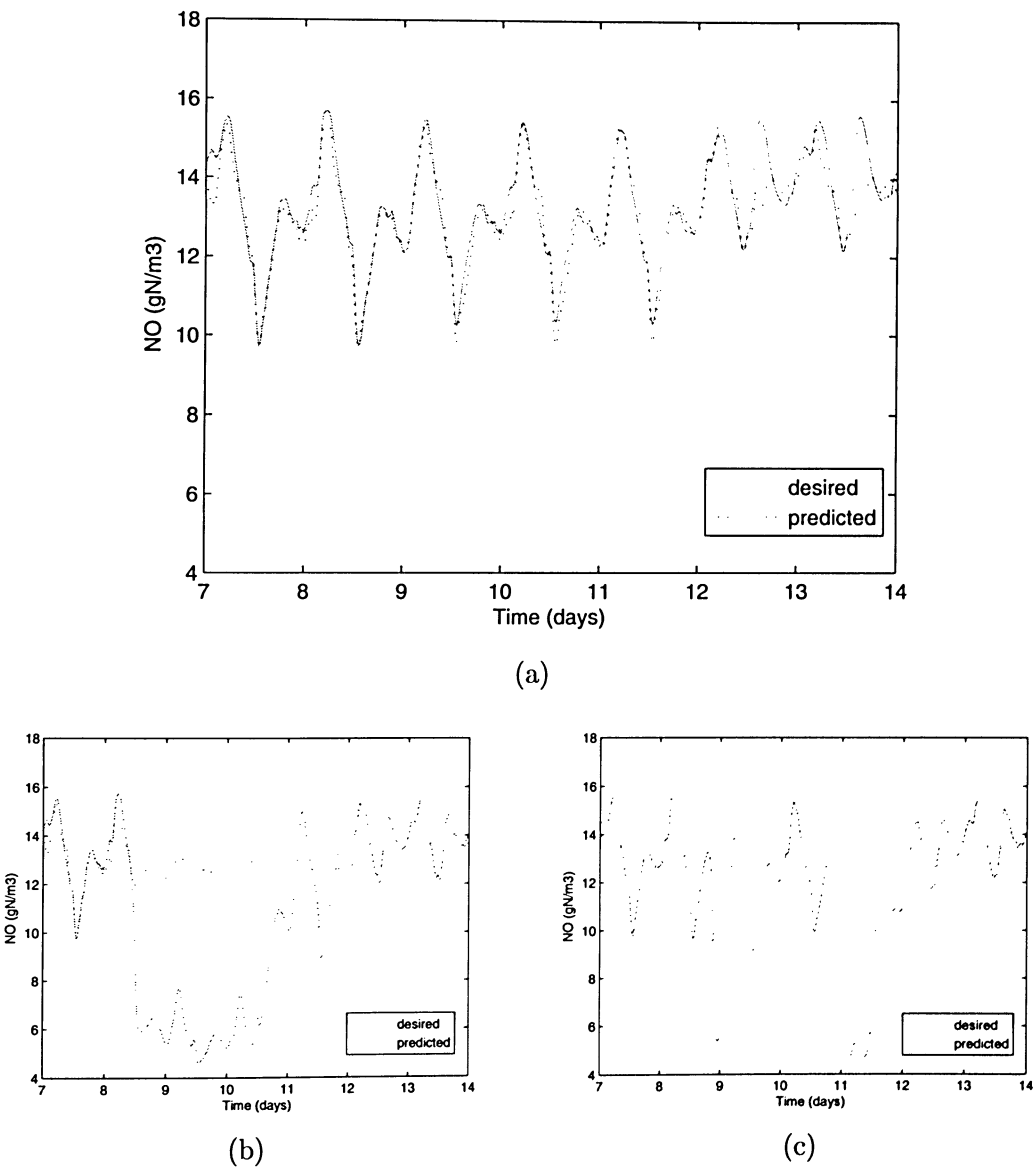


Figure 4.11: The comparison of predicted and desired NO concentration in ML under different weather conditions using GRNN (a) dry (b) rain (c) storm .

4.3.3 Comparison of the prediction results between LS-SVM and GRNN

In order to summarize the prediction performance of LS-SVM and GRNN and make their difference clear, we put together the simulation results with the optimal memory order in Table 4.5. As it is shown, the prediction performance of LS-SVM is better than that of GRNN, especially under the rain and storm event, which are not covered by the training data. These results reinforce the claim that SVMs have better generalization performance since they are based on the generalization bound derived from statistical learning theory.

Table 4.5: The prediction error comparison between LS-SVM and GRNN with the same input and output memory order.

Weather	**dry**	**rain**	**storm**
Memory order	Input: *3*, output:*1*	Input:*1*, output:*1*	Input:*1*, output:*1*
LS-SVM	0.1049	1.7146	2.0437
NN	0.3453	14.758	6.717

To have a rough idea about the speed for training and prediction of LS-SVM and GRNN, the simulations were run on an identical server. The server is IBM XSERIES_345 with 2.8GHz CPU and 3.75GB of RAM. The CPU computation time was recorded and listed in Table 4.6. In order to be fair, we didn't perform dimension reduction on both LS-SVM and GRNN. In our experiments, whether to use PCA doesn't affect the performance of GRNN significantly. However, we took the time needed for preprocessing in both LS-SVM and GRNN into account for the training time. As it is indicated in Table 4.6, the training and prediction of GRNN is a little faster than that of LS-SVM. But, considering the prediction errors compared in Table 4.5, we claim that the computation speed of LS-SVM is still in a reasonable range and the performance of LS-SVM is better than GRNN.

Table 4.6: The comparison of CPU computation time between LS-SVM and GRNN.

Learning machine	Weather	Training time	Prediction time
LS-SVM	dry	1.2813	8.4844
	rain	0.8281	7.6250
	storm	0.8125	7.6719
NN	dry	0.0781	5.6250
	rain	0.0781	5.1250
	storm	0.0625	5.0469

Chapter 5

Conclusions

In this thesis, first we reviewed the fundamental ideas and mathematical formulations of SVMs and NNs, with emphasis on LS-SVM, BPNN, RBFNN and GRNN. Then a benchmark nitrogen removal process in a wastewater plant was described. Both LS-SVM and GRNN were trained and tested on predicting NO concentration, with the following objectives:

- To investigate the prediction performance of LS-SVM
- To compare the generalization performance of LS-SVM and GRNN
- To test the sensitivity of memory orders of NARX model with both LS-SVM and GRNN

We will discuss the above three aspects with respect to the simulation results and then outline the future work.

5.1 Prediction performance of SVM

In a short term period, the LS-SVM model with RBF kernel can accurately predict the NO concentration in the effluent of the reactor, provided that the optimal parameters are selected. From the simulation results presented in Figure 4.10, the generalization ability of LS-SVM in combination with NARX model is evident. We used only the data representing dry weather condition to train LS-SVM, but we tested the prediction of NO concentrations under dry weather, rain and storm event, respectively. As we can see from the prediction

results, given the influent disturbances, the dynamics of the NO concentrations under dry, rain and storm weather conditions can be predicted using the same LS-SVM model trained by the same one-week dry weather data.

The generalization capability of LS-SVM can be attributed to the guaranteed upper bound of the generalization error by statistical learning theory.

To some extent, the excellent prediction performance of LS-SVM is also attributed to the using of the NARX model to extract the information from past data. The NARX model combined the power of regression and time series prediction. With the relatively optimal memory orders selected by experiments, NARX model shows effectiveness in transforming the input and output data into appropriate space in order to obtain better prediction results in the target problem.

5.2 Comparison of LS-SVM and GRNN

In order to compare the generalization performance of LS-SVM with that of GRNN, the comparative simulations were done using GRNN under the same settings and procedures, e.g., the same plant layout and specifications, same training data, same testing data and same memory orders in the NARX model.

As we used only the dry weather data to train the LS-SVM and GRNN, we want to see if the trained model can predict the NO concentration correctly under different weather patterns, such as rain or storm events. It can be seen from the prediction results shown in Figure 4.10 for LS-SVM and Figure 4.11 for GRNN, when predicting under dry weather, both LS-SVM and GRNN display comparatively good accuracy. However, the differences between the performance of LS-SVM and GRNN are obvious under the rain and storm events. We can see from the plots of influent flow rate (Figure 4.5) and NO concentration (Figure 4.8) under rain and storm event, as the influent flow increases, the NO concentration decreases. LS-SVM predictions reflected such patterns that were not contained in the dry weather data. In contrast, GRNN cannot predict such seemingly unusual pattern. This difference indicates that LS-SVM behaves well in prediction that requires extrapolation while GRNN can only be used to predict over the range of the training set.

We relate this difference of prediction performance between LS-SVM and GRNN to two points. As was explained, SVMs are based on the Structural risk minimization principle that consider to minimize the training error and the complexity of the model in a balanced way. In addition, as the statistical learning theory guarantees the generalization bound, the SVM-based methods like LS-SVM should have good generalization ability. In NN-based method, such as GRNN, the empirical risk minimization principle is applied and there is no guarantee for the generalization ability, especially when the training data don't contain all of the patterns that may exist within the entire data set.

5.3 Memory orders of NARX model

In this thesis, the NARX model was proposed to transform the inputs and outputs into a new state space in order to extract useful information. The embedding theory state that the forecasting performance could be seriously deficient if a model's memory order is either too little or too large [28]. Therefore, choosing the appropriate memory architectures for a given task is a critical issue in employing NARX models.

The sensitivity of the memory orders shows evident difference between LS-SVM and GRNN prediction.

From Table 4.3, it was shown that under dry weather the memory orders have no significant impact on the prediction performance for LS-SVM. But the memory orders do make difference on the prediction error under rain and storm events using LS-SVM.

As seen from Table 4.4, the memory orders in the NARX model have no significant impact on the prediction performance of GRNN under all three weather conditions.

In summary, when used for interpolation prediction (the training data and the testing data have the same pattern, e.g.,under dry weather), both LS-SVM and GRNN can perform well and the memory orders have little effects. When used for extrapolation (the training data and testing data have different patterns, e.g.,under rain and storm events), the memory orders have a strong impact on the prediction performance of LS-SVM and empirically selected optimal memory orders can help improve the generalization performance significantly. However, the memory orders have little effect and cannot improve the generalization

performance of GRNN in extrapolation.

These observations reinforce the power of the proposed solution by combining the LS-SVM with NARX model.

5.4 Future work

Since the simulation results gave us a strong evidence that LS-SVM combined with the NARX model can obtain a good generalization performance and prediction ability in the context of biological nitrogen removal processes, further study could emphasize on investigating and improving the performance of LS-SVM model in the case of real-world wastewater treatment plants. Some potential future work directions are listed as in the following:

1. Now that we have been successful with simulations, it is possible to investigate the performance of SVM in the case of real plants, using real data.

2. Using real data, we can carry out a comparative study of SVM and mechanistic models.

3. Develop a theoretical method to compute the memory orders automatically, in order to effectively extract the information from the past data.

4. Expand the applications of SVM to modeling the dynamics of other variables of wastewater treatment plants, including the phosphorous and the COD.

5. Study the generalization ability of SVM in the cases when data come from several different probability distributions.

6. Develop SVM-based control strategies for the plant using on-line learning.

Bibliography

[1] B. Baesens, S. Viaene, T. Van Gestel, J.A.K. Suykens, G. Dedene, B. De Moor and J. Vanthienen, *An Empirical Assessment of Kernel Type Performance for Least Squares Support Vector Machine Classifiers*, Fourth International Conference on knowledge-Based Intelligent Engineering systems & Allied Technologies, 2000.

[2] M.B. Beck, *Identification, estimation and control of biological wastewater treatment processes*, Proc. IEE 133, pp. 254–264, 1986.

[3] M.B. Beck, J.R. Ravetz, L.A. Mulkey and T.O. Barnwell, *On the problem of model validation for predictive exposure asesments*, Stochast,Hydrol. Hydraul., vol. 11, pp. 229–254, 1997.

[4] C.M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[5] B.R. Chang and S.F. Tsal, *Forecasting non-periodic short-term time series-radial basis function neural network approach*, IEEE SMC, vol. 6, pp. 4, 2002.

[6] J.B. Copp, *The COST Simulation Benchmark: Description and Simulator Manual. A Product of COST Action 624 and COST Action 682.* Directorate-General for Research, 2002.

[7] M. Cote, B.P.A. Grandjean, P. Lessard and J. Thibault, *Dynamic modelling of the activated sludge process: Improving prediction using neural networks*, Water Res., vol. 29(4), pp. 995–1004, 1995.

[8] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2003.

[9] R.-E. Fan, P.-H. Chen, and C.-J. Lin, *Working set selection using the second order information for training SVM*, Technical report, Department of Computer Science, National Taiwan University, 2005.

[10] GPS-X,[software] http://www.hydromantis.com/software02.html. [Accessed 01 April 2005].

[11] C.A. Gontarski, P.R. Rodrigues, M. Mori and L.F. Prenem, *Simulation of an industrial wastewater treatment plant using artificial neural networks*, Computers and Chemical Engineering, vol. 24, pp. 1719–1723,

[12] A. Guergachi and G.G. Patry, *Identification, Verification and Validation of Process Models in Wastewater Engineering: a Critical Review*, Journal of Hydro-informatics, vol. 5(3), pp. 181–188, 2003.

[13] A. Guergachi, *Integrating domain knowledge and machine learning theoretic methods for modelling complex environmental systems: methodology and rationales*, Proc. of the ISEIS 2003 International Conference on Environmental Informatics, Regina, pp. 386–393, 2003.

[14] M. Hack and M. Kohne, *Estimation of wastewater process parameters using neural networks*, Water Sci., vol. 33(1), pp. 101–115,

[15] M.F. Hamoda, I.A. Al-Ghusain and A.H. Hassan, *Integrated wastewater treatment plant performance evaluation using artificial neural networks*, Water Sci.Tech., vol. 40(7), pp. 55–65, 1999.

[16] F. Heimes and B. van Heuveln, *The Normalized Radial Basis Function Neural Network*, IEEE SMC, vol.2, pp. 1609 - 1614, 1998.

[17] M. Henze, Jr C.P.L. Grady, W. Gujer, G.v.R Marais and T. Matsuo, *Activated sludge model No.1.* IAWQ scientific and Technical Report No.1, IAWQ, London, 1986

[18] M. Henze, W. Gujer, T. Mino, T. Matsuo,M.C.Wentzel, and G.v.R Marais, *Activated sludge model No.2.* IAWQ Scientific and Technical Report, IAW Publishing, Copenhagen, Denmark, 1994.

[19] C.W. Hsu, C.C. Chang and C.J. Lin, *A Practical Guide to Support Vector Classification*, 2003.

[20] W. Gujer, M. Henze, T. Mino and M. van Loosdrecht, *Activated sludge model No.3*, Water Science and Technology, vol. 39(1), pp. 183–193, 1999

[21] U. Jeppsson, *Modelling Aspects of Wastewater Treatment Processes.* Ph.D. Thesis,Department of Industrial Electrical Engineering and Automation, Lund Institute of Technology, Lund, Sweden, 1996.

[22] T. Joachims, *Learning to Classify Text Using Support Vector Machines.* Dissertation, Kluwer, 2002.

[23] T. Joachims, *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods — Support Vector Learning, MIT Press, 1999.

[24] T. Joachims, *Transductive Inference for Text Classification using Support Vector Machines*, International Conference on Machine Learning (ICML), 1999.

[25] Ko Kang-Young, G.G. Patry, A.M. Cretu and E.M. Petriu, *Neural network model for wastewater treatment plant control*, IEEE International Workshop on Soft Computing Techniques in Instrumentation, Measurement and Related Applications, pp. 38–43, 2003.

[26] S. S. Keerthi and C.-J. Lin, *Asymptotic behaviors of support vector machines with Gaussian kernel*, Neural Computation, vol. 15(7), pp. 16671689, 2003.

[27] D.S. Lee and J.M. Park, *Neural Network Modelling for Online Estimation of Nutrient Dynamics in a Sequentially-Operated Batch Reactor*, Journal of Biotechnology, pp. 229–239, 1999.

[28] T.N. Lin, B.G. Horne, P. Tino and C.L. Giles, *Learning Long-Term Dependencies in NARX Recurrent Neural Networks*, IEEE Transactions on Neural Networks, vol. 7(6), pp. 1329–1338, 1996.

[29] T.N. Lin, C.L. Giles, B.G. Horne and S.Y. Kung, *A Delay Damage Model Selection Algorithm for NARX Neural Networks*, IEEE Transactions on Signal Processing, vol. 45(11), pp. 2719-2730, 1997.

[30] H.T. Lin and C.J. Lin, *A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods*, Technical report, Department of Computer Science and Information Engineering, National Taiwan University. http://www.csie.ntu.edu.tw/ cjlin/papers/tanh.pdf.

[31] LS-SVMlab, Least Squares - Support Vector Machines Matlab/C Toolbox, http://www.esat.kuleuven.ac.be/sista/lssvmlab/

[32] S. Marsili-Libelli, *Computer Control of the Activated Sludge Process*, Encyclopedia of Environmental Control Technology-Wastewater Treatment Technology, vol. 3, pp. 229-270, 1989.

[33] The Mathworks, *Neural Network Toolbox For Use with MATLAB*, User's Guide, version 4

[34] J. Mercer, *Functions of positive and negative type and their connection with the theory of integral equations*, Philos. Trans. Roy. Soc. London, 209, pp. 415-446, 1909.

[35] Metcalf and Eddy, Inc., *Wastewater Engineering: Treatment, Disposal, and Reuse.* 3rd Ed, McGraw-Hill, 1991.

[36] J.C. Platt, *Advances in Kernel Methods: Support Vector Machines*, B. Schlkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, Dec. 1998. Fast training of support ector machines using sequential minimal optimization.

[37] E.N. Sanchez, S. Celikovsky, J.M. Gonzalez and E. Ramirez, *Wastewater Treatment Plant Control by Combining Fuzzy Logic and Nonlinear Estimation*, Proceedings of the 2001 IEEE International Symposium on Intelligent Control, 2001.

[38] A.J. Smola, *Learning with Kernels*, Ph.D dissertation, GMD, Birlinghoven, Germany, 1998.

[39] A.J. Smola and B. Schölkopf, *A Tutorial on Support Vector Regression*, Royal Holloway College, London, U.K., NeuroCOLT Tech. Rep. TR 1998-030, 1998.

[40] B. Schölkopf, A. Smola, R.C. Williamson and P.L Bartlett, *New support vector algorithms*, Neural Computation, vol. 12(5), pp. 1083–1121, 2000.

[41] B. Schölkopf, Kah-Kay Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, *Comparing support vector machines with Gaussian kernels to radial basis function classifiers*, IEEE Transactions on Signal Processing, vol. 45(11), pp. 2758–2765, 1997.

[42] M. Shimakawa, S. Murakami and M. Mori, *Fuzzy modelling of an activated sludge process*, IEEE International Conference on Systems, Man, and cybernetics, vol. 2, pp. 12–15, 1999.

[43] J.C. Spall and J.A. Cristion, *A Neural network Controller for Systems with Unmodeled Dynamics with Applications to Wastewater Treatment*, IEEE Transactions on Systems, Man and Cybernetics, vol. 27(3), 1997.

[44] D.F. Specht, *A Generalized Regression Neural Network*, IEEE Trans. Neural Networks, vol. 2(6), pp. 568–576, 1991.

[45] J.A.K. Suykens, T. Van Gestel, J.De. Brabanter, B.De. Moor and J. Vandewalle, *Least Squares Support Vector Machines.* World Scientific, 2002.

[46] I. Takács, G.G Patry and D. Nolasco, *A dynamic model of the clarification thickening process*, Wat.Res., vol. 25(10), pp. 1263–1271, 1991.

[47] T.J. Terrillon, M.N. Shirazi, M. Sadek, H. Fukamachi and S. Akamatsu, *Invariant face detectionwith support vector machines*, Proc. of 15th International Conference on Pattern Recognition,Barcelona, vol. 4, pp. 210–217, 2000.

[48] V.N. Vapnik, *The nature of statistical learning theory.* New York:Springer-Verlag, 1995.

[49] V.N. Vapnik, *Statistical Learning Theory.* John Wiley & Sons, 1998.

[50] H. Vanhooren and K. Nguyen, "Development of a simulation protocol for evaluation of respirometry-based control strategies," *Technical Report*, University of Gent, Gent, Belgium 1996.

[51] W.L. Wang and M. Ren, *Soft-sensing method for wastewater treatment based on BP neural network*, Proceedings of the 4th World Congress on Intelligent Control and Automation, vol. 3, pp. 2330–2332, 2002.