SUPPORT VECTOR MACHINE AND ITS APPLICATIONS IN INFORMATION PROCESSING

By

VISHAL SAXENA

Bachelor of Technology, Indian Institute of Delhi, 2001
Master of Science, Georgia Institute of Technology Atlanta, 2003

Submitted to the Department of Civil & Environmental Engineering
In partial fulfillment of the requirements for the Degree of

MASTER OF ENGINEERING IN CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 2004

Signature of Author:
.....
Vishal Saxena
Department of Civil and Environmental Engineering
May 7, 2004

Certified by:
.....
John R. Williams
Associate Professor, Department of Civil & Environmental Engineering
Thesis Supervisor

Accepted By:
.....
Heidi Nepf
Chairman, Departmental Committee of Graduate Students

SUPPORT VECTOR MACHINE AND ITS APPLICATIONS IN INFORMATION PROCESSING

By

VISHAL SAXENA

Submitted to the Department of Civil & Environmental Engineering
On May 7th, 2004 in Partial fulfillment of the requirements for the Degree of the
Requirements for the Degree of Master of Engineering in
Civil & Environmental Engineering

ABSTRACT

With increasing amounts of data being generated by businesses and researchers there is a need for fast, accurate and robust algorithms for data analysis. Improvements in databases technology, computing performance and artificial intelligence have contributed to the development of intelligent data analysis. The primary aim of data mining is to discover patterns in the data that lead to better understanding of the data generating process and to useful predictions. One recent technique that has been developed to handle the ever-increasing complexity of hidden patterns is the support vector machine. The support vector machine has been developed as robust tool for classification and regression in noisy, complex domains. Current thesis work is aimed to explore the area of support vector machine to see the interesting applications in data analysis, especially from the point of view of information processing.

Thesis Supervisor: John R. Williams

Title: Associate Professor, Department of Civil & Environmental Engineering

ACKNOWLEDGEMENTS

I dedicate this thesis to my family for providing me with inspirations and encouragements while at MIT.

Firstly, I would like to thank my parents for all their sacrifice, support and hard work to get me to where I am today. I am thankful to my fiancée, Rashmi Srivastava for her encouragement and support throughout my entire stay in Boston.

Secondly, I would like to thank my thesis advisor Dr. John R. Williams, for his constant guidance, advice and encouragement throughout my stay here at MIT. His support, suggestion and comment have been very vital for the completion of this thesis work. It is my great privilege and honor to have had the opportunity of working with him for the past one year.

Thirdly, I would like to give many thanks to all the faculties, students and administrative staffs at the Department of Civil & Environmental Engineering for making my education a valuable learning experience.

Lastly, but most importantly, I would like to thank God for blessing me with the opportunity to pursue my education this far.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. OVERVIEW

With increasing amounts of data being generated by businesses and researchers there is a need for fast, accurate and robust algorithms for data analysis. Improvements in databases technology, computing performance and artificial intelligence have contributed to the development of intelligent data analysis. The primary aim of data mining is to discover patterns in the data that lead to better understanding of the data generating process and to useful predictions. Examples of applications of data mining include detecting fraudulent credit card transactions, character recognition in automated zip code reading, separating Spam from regular emails and predicting compound activity in drug discovery. Real-world data sets are often characterized by having large numbers of examples, e.g. billions of credit card transactions and potential 'drug-like' compounds; being highly unbalanced, e.g. most transactions are not fraudulent, most compounds are not active against a given biological target; and, being corrupted by noise. The relationship between predictive variables, e.g. physical descriptors, and the target concept, e.g. compound activity, is often highly non-linear. One recent technique that has been developed to address these issues is the support vector machine. Unlike traditional methods which minimize the empirical training error. SVM aims at minimizing an upper bound of the generalization error through maximizing the margin between the separating hyper plane and the data. This can be regarded as an approximate implementation of the Structure Risk Minimization principle. What makes SVM attractive is the property of condensing information in the training data and providing a sparse representation by using a very small number of data points. SVM is generally used to find linear classifier and takes care of non-linearity by transforming input data to feature space. By choosing

an adequate mapping the data points become linearly separable or mostly linearly separable in the high dimensional space so that one can easily apply the structure risk minimization. We need not compute the mapped patterns explicitly and instead we only need the dot products between mapped patterns. They are directly available from the kernel function which generates such mapping patterns. By choosing different kinds of kernels, SVM can realize Radial Basis Function Polynomial and Multi-layer Perceptron classifiers. Compared with the traditional way of implementing them SVM has an extra advantage of automatic model selection□ in the sense that both the optimal number and locations of the basis functions are automatically obtained during training The support vector machine has been studied as robust tool for classification and regression in noisy, complex domains. The two key features of support vector machines are generalization theory, which leads to a principled way to choose a hypothesis; and, kernel functions, which introduce non-linearity in the hypothesis space without explicitly requiring a non-linear algorithm.

Current thesis work is aimed to explore the area of support vector machine to see the interesting applications in data analysis, especially from the point of view of information processing. The history of learning algorithm and critical advancements in the area of SVM will be presented. Mathematical formulation of basics for the binary classification algorithm will be studied. Algorithm will be adapted to work for various applications of data refining and data mining and computer implementation of such modified algorithms will be used to test the validity of its workings.

One of the interesting data classification problems that will remain the main focus of current thesis work is problem of signature recognition. Establishing the identification of

a signature by automatic search through a database of signatures is of interest in several areas. However, as real-world applications for signature recognition system continues to increase, the need for an accurate, easily trainable recognition system becomes more demanding. Current systems have advanced to be fairly accurate in recognition under constrained scenarios, but extrinsic identification parameters such as the speed at which signature is written, occasional carelessness in signature creation, and the accuracy of writing instrument still cause much difficulty in correct recognition.

In the current research work, a method for signature recognition through the use of time series generation and Support Vector Machine is being proposed. First, a time series for a given signature at run time will be extracted. For a given person, numerous numbers of signatures producing different time series will be used to generate a set of attribute vector. These vectors will be the basis for training the Support Vector Machines. Once SVM is trained, any new test signature will be processed to generate a vector of attributes and resulted vector then can be tested using trained SVM to find whether this new signature belong to the person for which SVM was tested or not.

2. INTRODUCTION

Support Vector Machines (SVM) are learning systems (algorithms) that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. They have been trained to perform binary classification (pattern recognition) and real valued function approximation (regression estimation) tasks. SVM being a supervised learning requires learning machine to scan through a given training set of examples with associated labels. Usually the examples are in the form of attribute vectors, so that the input space is a subset of Rn. Once the attributes vectors are available, a number of sets of hypotheses could be chosen for the problem. Among these, linear functions are the best understood and simplest to apply.

In general, complex real-world applications require more expressive hypothesis spaces than linear functions. Another way of viewing this problem is that frequently target function cannot be expressed as simple linear combination of the given attributes, but in general requires that more abstract features of the data be exploited. Multiple layers of threshold linear functions were proposed as a solution to this problem, and this approach led to the development of multi-layer neural networks. Kernel representations offer an alternative way of projecting the data into a high dimensional feature space to increase the computational power of the linear support vector machines. Choice of such appropriate kernel functions will be studied in great depth in current thesis work.

Importantly, a dual form of the optimization problem involved in SVM gives rise to some very important results like not all the data points contribute to the choice of best

classification function. In subsequent chapters, we will look at number of similar directly applicable concepts that result from theoretical setup of the SVM.

3. CLASSICAL CLASSIFICATION TECHNIQUES

The design of machines capable of learning from experience has for a long time been the moot point for both philosophical and technical debate. The technical aspect of machine learning has received an enormous growth from the advent of modern computers. They have demonstrated that machines can display a significant level of learning ability, though the boundaries of this ability are far from being clearly defined. The availability of reliable learning algorithm is of strategic importance, as there are still lots of tasks that have not been solved by computer programming, since no definite mathematical formulation for these tasks is available. However if we have a look at the history of the generation of learning algorithms, Pre 1980 phase saw that almost every learning methods learned linear decision surfaces and such linear learning methods had nice theoretical properties. In the decade following 1980, decision trees and neural networks allowed efficient learning of non-linear decision surfaces. There was little theoretical basis and all suffered from local minima. The 1990 brought efficient learning algorithms for non-linear functions based on computational learning theory developed. Once again, such algorithms had nice theoretical properties. However two independent developments within last decade, computational learning theory and new efficient separable non-linear functions that use "kernel functions" brought this new idea of Support Vector Machine. This resulting learning algorithm is an optimization algorithm rather than a greedy search

3.1.Supervised Learning

A ubiquitous problem in statistics and lately in computer science with applications in many areas is to guess or estimate a function or a relationship from some example input-output pairs with little or no knowledge of the form of the function. So common

is the problem that it has different names in different disciplines (e.g. nonparametric regression, function approximation, system identification, inductive learning).

In neural network nomenclatures, the problem is called supervised learning. The function is learned from the examples, which a teacher supplies. The set of examples, or training set, contains elements, which consist of paired values of the independent (input) variable and the dependent (output) variable. For example, the independent variable and dependent variable might exist in following mathematical form:

Where vector contains say n attributes like and may take any real value. Technically, such problems look for some sort of decision functions that can generate output value by looking at a vector of attribute.

Machine learning algorithms receive examples from a teacher or from the environment and attempt to learn some structure that will generalize for unseen examples. The majority of work in Machine Learning falls into three learning frameworks: supervised, unsupervised, and reinforcement.

Machine learning algorithms receive examples from a teacher or from the environment and attempt to learn some structure that will generalize for unseen examples. The majority of work in Machine Learning falls into three learning frameworks: supervised, unsupervised, and reinforcement.

In supervised learning, the algorithm attempts to learn a concept from labeled examples that predicts the labels of the training examples correctly and generalizes to produce correct labels on examples outside of the training set. The label can be a class, in which case the learning task is called classification, or a continuous signal, in which case the task is called regression. Some concept representations used to learn

from labeled examples include decision trees [Quinlan1992], nearest neighbor [Dasarathy1991], neural networks [Rumelhart1986], and Bayesian networks [Pearl1988].

In unsupervised learning, the examples are not labeled. The algorithm attempts to learn the structure of the underlying source of the examples. Some instances of unsupervised learning are clustering [Cheeseman1988], Principal Component Analysis [Chatfield and Collins1980], and Independent Component Analysis [Bell and Sejnowski1995, Comon1994].

In Reinforcement Learning the goal is to learn a policy, which is a mapping from states to actions. Examples are not labeled with the correct action; instead an occasional reinforcement signal, which denotes the utility of some state, is received. Reinforcement Learning methods are reviewed in [Sutton and Barto1998] and [Kaelbling 1996].

3.2. Classification Problem

If output variable take finite number of real values, problem can be seen as classification problem, where one is trying to see which category some given vector of attribute (input variable) maps to? If output variable is bound to have only binary values, problem falls into category of Binary Classification. Here, any given input variable will either belong to some fixed category or it does not. It is irrelevant to see what happens to input variable if it does not fall into that fixed category, this kind of problem looks like "choice of good and bad". Either something is good or otherwise it is bad.

Binary classification is aimed to find out the real valued function that can separate input data in two categories, A and B or +1 & -1 or 1 & 0 or good or bad. The

function to be found out is termed as classification function or decision function or hypothesis. Clearly, output variable takes only two values.

As seen earlier, input variable in general is a vector and has n attributes. The choice and estimation of attribute will depend on the problem itself and will be discussed while focusing on various applications towards the last chapters of thesis work.

Combining above-mentioned nature of input and output, decision functions can be seen as following mapping:

$$f : X \subseteq R^n \rightarrow R \text{ ---------- (1)}$$

Entire objective of classification problem is to be able to find this function.

3.3.Linear Learning Machines

Linear learning machines are those computer programs that focus on the linear decision function. These are the most basic form of learning machines and are easiest to be used as a platform for building concepts of Support Vector Machines in subsequent chapters.

Mathematically, linear learning machines can be represented as:

$$f(x) = \langle w.x \rangle + b \text{ ---------- (2)}$$

$$f(x) = \sum_{i=1}^{n} w_i.x_i + b \text{ ---------- (2a)}$$

In two-dimensional case, input variable is a vector of two attributes and hence decision function is a linear function of two variables, or in other words it is simply a line. The equation of line is the only solution for linear learning machines trying to operate on two attributes vector input variable and in multi-dimensional case it becomes a hyper-plane. The equation 2 shows a dot product of equation of a hyper-plane and input vector. The equation of hyper-plane is split in the coefficients vector

w (also called as weight vector) and a constant term b. Dimensionality of input vector will decide the dimensionality of hyper-plane, so if we are dealing with n attributes vector, hyper-plane will be based in an n-dimensional space and will require n coefficients and a constant term. (As an example a.x + b.y + c is a line where a & b are coefficient terms and c is constant). The vector w is the vector of these n coefficients.

If function once operated on input variable gives a positive value, input variable falls into one category (say A or +1) otherwise it falls into the other one (say NOT A or –1). This is one of the different ways to categorize data into two classes. Whether we agree on the sign of output or compare the output relative to some fixed real number, classification remain the same as it is simply a matter of changing shifting or equation of hyper-plane. However, sign of the output as the basis for category identification is the easiest to put into mathematics and all the formulation in this thesis will be on this basis.

$$f(x_i) \geq 0 \Rightarrow x_i \in A, y_i = +1 \text{ ---------- (3)}$$

$$f(x_i) < 0 \Rightarrow x_i \in A, y_i = -1 \text{ ---------- (3a)}$$

Combining equation 3 and 3a, we arrive at

$$y_i f(x_i) \geq 0 \text{ ---------- (4)}$$

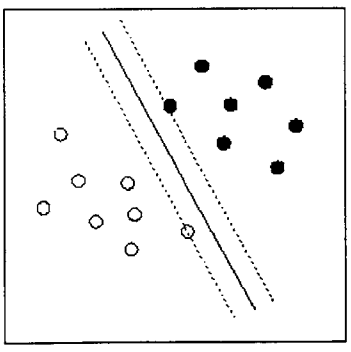

Figure 1: A hyper-plane separating data into two categories

A geometric interpretation of this classification approach is that data is being separated by a hyper-plane. Mathematically, vector w shows the direction perpendicular to this hyper-plane.

Now, how do we find this hyper-plane is still a question to be addressed? The approach to arrive at best hyper-plane is the core essence of entire classification algorithm research. We will start with one of the most popular traditional algorithm, so-called Perceptron Algorithm.

3.4.Perceptron Algorithm

Rosenblatt developed the perceptron algorithm in 1956. It is an "on-line" and "mistake-driven" algorithm, which starts with some initial assumption of weight vector $w_o \& b_o$. Usually, a zero vector is taken for initial approximation of weight vector and constant term is also started with zero. Each time a training point is misclassified by current hyper-plane ($w \& b$), algorithm makes a revision to classification function and tests all the training points with respect to revised hyper-plane. It is a iterative process that continue till all the training points satisfy some classification function. Depending on the distance between approximation and actual hyper-plane and on the direction in which algorithm proceeds, it may take large number of iterations and at times may prove to be impractical for all practical purposes. If data is non-separable by a linear function, this algorithm will be in infinite loop for sure.

Given a linearly separable training set S and learning rate $\eta \in R^+$

$w_o \leftarrow 0; b_o \leftarrow 0; k \leftarrow 0$

$R \leftarrow \max_{1 \le i \le l} \|x_i\|$

repeat

for i = 1 to l

if $y_i(\langle w_k . x_i \rangle + b_k) \le 0$ then

$w_{k+1} \leftarrow w_k + \eta y_i x_i$

$b_{k+1} \leftarrow b_k + \eta y_i R^2$

k = k+1

end if

end

until no mistakes made within the for loop

return (w_k, b_k) where k is the number of mistakes

Table 1: The perceptron algorithm

3.5.Implementation

In an effort to see the working of traditional neural net approach of learning systems, Perceptron algorithm has been implemented in C# and tested here. The program is attached with this thesis work. It is a console based application that asks for the dimension of input space or number of attributes in a training example. It also require user to feed in the number of test data and then a learning parameter. It then follow algorithm described in Table 1 to find an appropriate hyper-plane. The moment it arrives at some hyper-plane that satisfy all the training data, it give it back to user.

3.6.Test Run

Here is the test run that try to find a separating hyper-plane for two dimensional data.

Test Run: - Here is the excel plot of the results obtained by console based application for another set of two dimensional training examples.

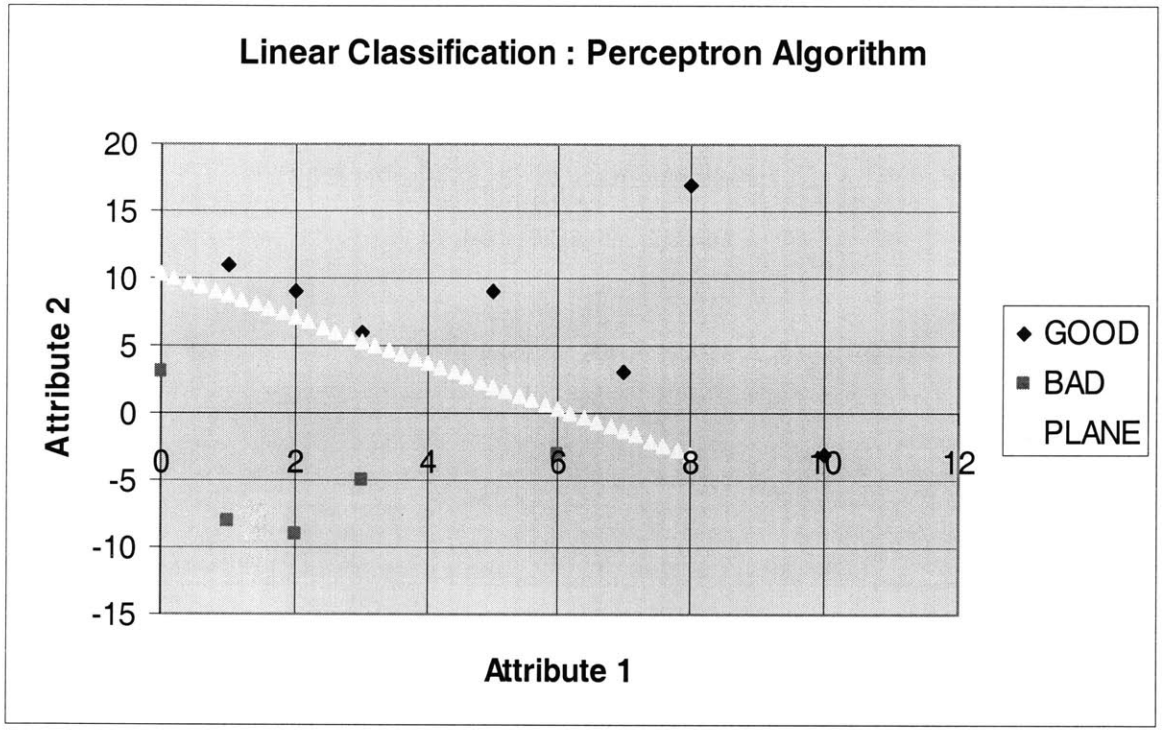

Figure 2: A hyper-plane from the Perceptron algorithm

3.7.Limitations

One of the major disadvantages with this kind of learning is the fact that it does not give the best answer. As and when it find first separating hyper-plane it stops here. In fact there is no way by which this algorithm can find all the hyper-plane and then pick up the best one. Choosing the best hyper-plane require some sort of criteria, which was absent in this kind of algorithm. It can not cover all the hyper-planes as it don't iterate over test examples in some defined and logical way. This algorithm is basically a blind search for anything that just satisfy all the test runs

More importantly, the order in which test run examples are fed to algorithm makes a difference in the result. This is because of the fact that trial of some of the examples can force algorithm to converge earlier than other ones. Some of the test examples may simply sit closer to final answer than others.

This algorithm does not guarantee any convergence as there might be a set of data that is simply not classifiable using any hyper-plane. This kind of data is known as non-separable by liner hyper-plane and cause algorithm to traverse entire space indefinitely

The test runs of Perceptron Algorithm motivates us to find much more intelligent way of finding a separating hyper-plane where all possible solutions can be analyzed and can be optimized to produce the best solution based on some criteria. This kind of learning will not be only very effective but also will carry more sense as it brings much more factor in light that can be configured depending upon necessities of application to application.

Support Vector Machine is a learning machine that fulfill most of the need that have been talked about in this section and is a much more mathematically robust and organized way of extracting hidden information from the example data. Before we delve into details of SVM, first conventions needed to support theory of SVM will be described.

4. CONVENTIONS

From chapter 2, linear learning machine can be mathematically written as:

$$f(x) = \langle w.x \rangle + b \text{ ---------- (2)}$$

$$f(x) = \sum_{i=1}^{n} w_i.x_i + b \text{ ---------- (2a)}$$

As discussed earlier w and b collectively define a hyper-plane and termed as weight vector and bias. The function f is the decision function or classification function as shown in equation 1.

We define functional margin of an example (x_i, y_i) with respect to a hyper-plane (w,b) to be the quantity

$$\gamma_i = y_i(\langle w.x_i \rangle + b) \text{ ---------- (5)}$$

Note that as per equation 3 and 3a $\gamma_i > 0$ means correct classification of (x_i, y_i) by decision function f.

The functional margin distribution is the distribution of the functional margin of the examples in space S for some given classification function f. Minimum of functional margin distribution for a given set S and classification function f or hyper-plane (w,b) is known as functional margin of hyper-plane or classification function with respect to data set S. If (w,b) is normalized by dividing itself by its norm $\|w\|$, functional margin results into geometric margin, functional margin distribution turns out to be geometric margin distribution and functional margin of hyper-plane gives functional margin of normalized

hyper-plane $(\frac{w}{\|w\|}, \frac{b}{\|w\|})$ or called its geometric margin over set S. Margin of entire set S is the maximum geometric margin over all possible normalized hyper-planes $(\frac{w}{\|w\|}, \frac{b}{\|w\|})$.

5. MATHEMETICAL TOOLS

Basic mathematical formulation that will be presented in chapter5, involves quite a robust mathematical analysis, particularly manipulation of optimization problem between its primal and dual form and its solution through Lagrange's Multiplier method.

Therefore, this chapter is an effort to touch the base of these tools to make formulation of SVM in next chapter a bit easier.

5.1.Optimization Theory

The problem of maximizing or minimizing a function against a set of constraints is called an optimization problem and has been topic of intense research since years in the area of operation research. The function to be optimized is known as objective function. Constraints enforced on optimization may be equality or inequality constraints. The set of variable that results into optimal value of objective function is known as solution of optimization problem.

An optimization problem that has linear constraints and linear objective function is known as Linear Programming and is briefed as LP.

$$Max\{Ax : a_i^T x \le b_i, a_j^T x = b_j\}, i \,\&\, j \in [0,n] \text{---------- (6)}$$

The quadratic programming problem involves minimization of a quadratic function subject to linear constraints. Most general formulation can be described as:

$$Min\{\frac{1}{2}x^T Qx + c^T x : a_i^T x \le b_i, a_j^T x = b_j\}, i \,\&\, j \in [0,n] \text{---------- (7)}$$

5.2.Primal & Dual

As a general rule of thumb, any maximization problem can be converted into a minimum problem or vice versa. A LP primal can easily be described in its primal and dual form

Primal:

$Max\{C^T x : Ax \leq b : x \geq 0\}$ ---------- (8)

Dual:

$Max\{b^T y : A^T y \geq C : y \geq 0\}$ ---------- (8a)

In fact, following table can be used for interchanging any primal into its dual or vice versa. In order to interchange primal and dual, change constraints and operators as shown in the table.

Primal	**Dual**
$\leq$ Constraint	Variable ≥ 0
$\geq$ Constraint	Variable ≤ 0
= Constraint	Variable unrestricted
Variable ≥ 0	$\geq$ Constraint
Variable ≤ 0	$\leq$ Constraint
Variable unrestricted	= Constraint
Max	Min
RHS	Objective coefficients
Objective coefficients	RHS
No. Of constraints	No. Of variables
No. Of variables	No. Of constraints

Table 2: Primal Dual Conversion

A quadratic programming problem described as:

$Min\{\frac{1}{2} w^T Q w - k^T w : Xw \leq c\}$ ---------- (9)

Where Q is a positive definite n-by-n matrix, k is an n-vector, c an m-vector, w the unknown and X being m by n matrix. This primal can be viewed in its dual form as

$Max\{-\frac{1}{2}\alpha^T P \alpha - \alpha^T d - \frac{1}{2} k^T Q k : \alpha \geq 0\}$ ---------- (9a)

Where $P = XQ^{-1}X^T$ and $d = c - XQ^{-1}k$. It is important to see that dual of a quadratic problem gives rise to another quadratic problem but we constraints being a bit simpler. This idea will be leveraged while defining the idea of support vectors in upcoming chapters.

5.3.Lagrange's Multiplier

This is the most important and widely used technique for solving optimization problem and will play a vital role in deriving basic mathematical formulation of SVM. This method was developed by Lagrange in 1797 in order to efficiently arrive at a solution of an optimization problem initially when there are no inequality constraints.

In general, Lagrange multipliers are used to find the extreme points of $f(x_1, x_2 \cdots x_n)$ subjected to constraints $g(x_1, x_2 \cdots x_n) = C$, where f and g are functions with continuous first partial derivatives on the open set containing the curve $g(x_1, x_2 \cdots x_n) = 0$, and $\nabla g \neq 0$ at any point on the curve. Here operator ∇ is well-defined gradient operator:

$$\nabla = \frac{\partial}{\partial x_1}\hat{x}_1 + \frac{\partial}{\partial x_2}\hat{x}_2 + \cdots + \frac{\partial}{\partial x_n}\hat{x}_n \text{ ---------- (10)}$$

In order to maximize or minimize a function, its partial derivatives with respect to every single variable is evaluated and equated to zero.

Mathematically, it can be expressed as:

$$\nabla f . d\hat{x} = \frac{\partial f}{\partial x_1}dx_1 + \frac{\partial f}{\partial x_2}dx_2 + \cdots + \frac{\partial f}{\partial x_n}dx_n = 0 \text{ ---------- (11)}$$

More importantly, equation 11 gives rise to n different equations as infinitesimal change in any dimension is an independent variable.

$$\frac{\partial f}{\partial xi} = 0 \forall 0 \leq i \leq n \text{ ---------- (12)}$$

Now, partially differentiating constraint function with respect to each of the variable we arrive at:

$$\frac{\partial g}{\partial xi} = 0 \forall 0 \leq i \leq n \text{ ---------- (13)}$$

Combining equation 11 and equation 12, we arrive at

$$\frac{\partial f}{\partial xi} + \lambda \frac{\partial g}{\partial xi} = 0 \forall 0 \leq i \leq n \text{ ---------- (14)}$$

The parameter λ is known as Lagrange's multiplier and can be treated as a new variable for each of the constraint that we have.

In fact, optimization function once combined with the constraints through this parameter proves to be another optimization function which only need to be optimized with respect to all original variables and at the same time with respect to every Lagrange multiplier that have been added because of some constraint. In this process we got rid of constraints and are able to focus only on one objective function.

In short, objective function becomes

$$f(x_1, x_2 \ldots x_n) + \lambda(g(x_1, x_2 \ldots x_n) - c) \text{ ---------- (15)}$$

Or in case of multiple constraints,

$$f(x_1, x_2 \ldots x_n) + \lambda_1(g_1(x_1, x_2 \ldots x_n) - c_1) + \lambda_1(g_1(x_1, x_2 \ldots x_n) - c_1) + \cdots \text{ ---------- (16)}$$

6. KERNEL FUNCTIONS

The complexity of the classification function imposes a challenge to machine learning. In real world applications, patterns are so deep hidden that it requires sophisticated data mining techniques to dig them out. Even if the patterns are simpler, the space in which data exist might be inappropriate to see the hidden patterns. Consider following example

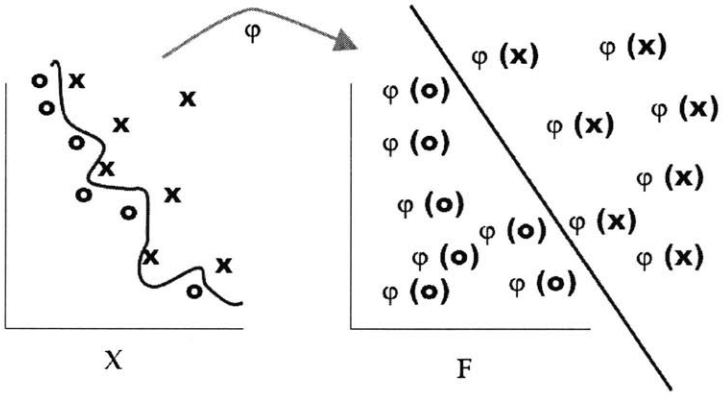

Figure 3: Kernel Mapping from Input Space to Feature Space

A researcher is looking at a data that he or she thinks would be appropriate to be separates in two categories by a line. As shown in figure 2, left hand side picture shows that data can be separated in two different categories (circle and cross) by a wiggly line but cannot be by a straight line. If research keeps finding a straight line in this space as it is, it is just next to impossible to be able to separate this data into circles and crosses. But now, consider that researcher process this data through some sort of transformation and is able to present it as shown in right hand side image of figure 2. It is so easy to draw a line now!

This is just merely a fact that some relationship may not be seen in the input domain of data as it is. But if data can be transformed into some other domain; it might be possible to be extract the required relationship from it. Continuing same discussion to a level

higher, dimension of the input space at times might also pose similar challenges and need to be changed in order to see the same data in some form. Support Vector Machine explores this idea of transforming the input domain into a higher dimensional space to be able to optimize over best of the best classification functions which otherwise will be impossible to realize.

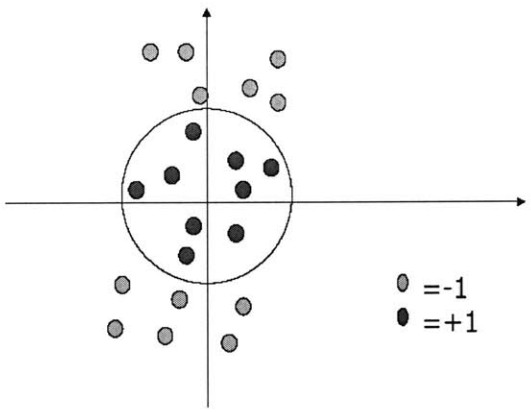

Figure 4: Non-linear separable data

As a next example of showing the power of kernel functions, consider the data points as shown in figure 3. This data can not be separated if a line is decided to be a separating hyper-plane. Clearly data is non-linear. But it looks like for sure that red circles and green circles are not inter-mingled and hence should be separated in two categories. Here choice of a line as a separating hyper-plane is wrong. If we choose a circle to separate them, it will work quite nicely.

Subsequent chapters show that a linear SVM is quite complicated from mathematics point of view but relatively easier as compared to their non-linear counterparts. Linear machines require linear classification functions or lines in 2-D case. Now if we process data shown in figure 3 by adding one more dimension say z and assigning z values of

every data point to be +1 or -1 depending upon they lie in a circle of pre-calculated radius or nor. A closer look in this 3 dimensional feature space shows that z=0 is a hyper-plane that will separate data into red and green circles. So by exploiting power of some sort of transformation function functions we are able to use linear functions to separate a data that was next to the impossible to be separated in input space because of its non-linear nature. This example also brings an important point in discussion that is choice of an appropriate transformation functions require some sort of knowledge with data before hand. The proper selection is core to the art of SVM.

The functions that transform the input domain to appropriate domain are called *Kernel Functions*. The new space or domain is called *Feature Space*. Feature space makes us more comfortable with data and helps us retrieving patterns that can never be seen in input space. Hence in short, Kernel functions are way of transforming input space to feature space. The kernel parameters and the kernel function itself are some of the few tunable parameters in support vector machines. Their choice drastically influences the quality of the returned results.

Some of the most popular kernel functions that have been frequently used are:

Linear Kernel is: $K(x,z) = \langle x.z \rangle$ ---------- (17)

Polynomial Kernels are: $K(x,z) = (\lambda \langle x.z \rangle + c)^d$ ---------- (17a)

Radial basis functions are: $K(x.z) = \exp(-\gamma \|x - z\|^2$ ---------- (17b)

The kernel functions as used in equation 17 operate on two vectors rather than transforming one vector at a time. This will be explained a little further once we dig out the proper place to plug in these functions in the formulation of learning machines.

Since learning is being done in feature space rather than input space so equation 2 needs slight modification. Instead of using input vector x, we use transformed vector $\phi(x)$

$$f(x) = \langle w.\phi(x) \rangle + b \text{ ---------- (18)}$$

$$f(x) = \sum_{i=1}^{N} w_i \phi_i(x) + b \text{ ---------- (18a)}$$

The equation 18a simply shows that instead of operating on input vector first and then extracting every single attribute of feature vector while taking its dot product with weight vector, we can extract all the attributes of feature vector by visualizing a different kernel function for every single attribute and operating on attribute to attribute basis. Following example will make this difference clear:

Suppose $x = (x_1, x_2)$ and transformation required is $(x_1, x_2) \rightarrow \phi(x_1, x_2) = (x_1^2, x_2^2, x_1 x_2)$

Here, input vector is first transformed from a 2-D space to 3-D space and now all these three attributes of feature vector will be multiplied with corresponding weights of hyper-plane as according to equation 18.

We could also have done it directly by assuming three kernel functions for extracting three different attributes like:

$$\phi_1(x) = x_1^2, \phi_2(x) = x_2^2, \phi_3(x) = x_1.x_2$$

In general, it will always be beneficial to project data in a higher dimensional space as it removes coupling between data and make SVM work out a linear classification function.

7. SUPPORT VECTOR MACHINE

7.1.Basics

The aim of Support Vector classification is to devise a computationally efficient way of learning optimal separating hyper-planes in a high dimensional feature space. The simplest model of Support Vector Machine, which was also the first to be introduced, is the so-called maximal margin classifier. It works only for the data that is linearly separable in some feature space; the choice of appropriate kernel functions is still a moot point. Not only it is the easiest algorithm to understand but also provides most of the insight that is needed to form the building blocks of any Support Vector Machine.

One of the factors that give rise to birth of different SVM is the basis of the optimal hyper-plane. Maximal margin classifier considers the optimality based on the margin of a training set. It picks up the hyper-plane that corresponds to the maximum margin of training set. The geometric interpretation behind this basis is that it chooses the hyper-plane that look for a gap separating data in two parts and offers maximum gap over all possible hyper-planes. (This all happens in feature space!)

7.2.Convex Optimization Problem

Mathematically, problem of finding maximal margin hyper-planes can be seen as a convex optimization problem: minimize a quadratic function under linear inequality constraints.

As defined in Chapter 3, the geometric margin of a training point $(x_i, y_i) \in i = 1 \cdots l, x_i \in R^N, y_i \in \{-1,+1\}$ with respect to a hyper-plane (w,b) is given by

$$\gamma_i = y_i(\langle w.x_i \rangle + b) \text{ ---------- (5)}$$

Where hyper-plane should be normalized hyper-plane $(\frac{w}{\|w\|}, \frac{b}{\|w\|})$. After normalization equation 5 results into:

$$\gamma_i = y_i(\langle \frac{w}{\|w\|} . x_i \rangle + \frac{b}{\|w\|}) \text{ ---------- (19)}$$

The geometric margin of the training set S with respect to hyper-plane (w,b) is the minimum among all the training points. In order to further simplify this problem, functional margin of training set S is kept at unity and norm is minimized so that geometric margin of entire training set S is maximum at $\frac{1}{\|w\|}$.

This assumption of unit functional margin imposes following constraints:

$$\langle w.x^+ \rangle + b \geq +1, \langle w.x^- \rangle + b \leq -1 \text{ ---------- (20)}$$

$$y_i(\langle w.x^+ \rangle + b) \geq +1 \text{ ---------- (20a)}$$

Here, $.x^+$ and $.x^-$ are two set that hyper-plane is able to separate out. The equation 20 simply says that functional margin of entire training set with respect to a fixed hyper-plane is at least +1 if it is in +1 category $(y_i = 1)$ otherwise it is -1 $(y_i = -1)$. The equation 20a incorporates the value of y_i in equation 20.

In order to maximize geometric margin of a hyper-plane, while keeping function margin at unity, norm of the hyper-plane (denominator of geometric margin) need to be minimized. Therefore, we arrive at following optimization problem:

$$Minm\{\|w\| : y_i(\langle w.x^+ \rangle + b) \geq +1\} \text{ ---------- (21)}$$

Solving equation 21 is the sole objective of SVM based on Maximal Margin Hyper-plane.

The primal using Lagrange's multiplier can be written as shown below. Minimize $L(w,b,\alpha)$, where $L(w,b,\alpha)$ concatenates objective function and l constraints through l Lagrange's multipliers $\alpha \geq 0$.

$$L(w,b,\alpha) = \frac{1}{2}(w.w) - \sum_{i=1}^{l} \alpha_i [y_i(\langle w.x_i \rangle + b) - 1] \text{ ---------- (22)}$$

Transforming this optimization problem into its dual form is the next step. Partially differentiating $L(w,b,\alpha)$ with respect to w, b and α_i equating them to zero:

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{l} y_i \alpha_i x_i = 0 \text{ ---------- (23)}$$

$$w - \sum_{i=1}^{l} y_i \alpha_i x_i = 0 \text{ ---------- (23a)}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0 \text{ ---------- (24)}$$

$$\sum_{i=1}^{l} y_i \alpha_i = 0 \text{ ---------- (24a)}$$

Substituting results back into equation 22, we arrive at

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2}(w.w) - \sum_{i=1}^{l} \alpha_i [y_i(\langle w.x_i \rangle + b) - 1] \\ &= \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle x_i.x_j \rangle - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle x_i.x_j \rangle + \sum_{i,j=1}^{l} \alpha_i \text{ ---------- (25)} \\ &= \sum_{i,j=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle x_i.x_j \rangle \end{aligned}$$

Dual becomes,

$$\max\{W(\alpha) = \sum_{i,j=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle x_i.x_j \rangle : \sum_{i=1}^{l} y_i \alpha_i = 0, \alpha_i \geq 0, i = 1 \cdots l\} \text{ ----- (26)}$$

If multipliers α_i' and weight vector w'' are the optimal solutions to equation 26, they will be satisfying following constraints as per equation 23a:

$$w' = \sum_{i=1}^{l} y_i x_i \alpha_i' \text{ ---------- (27)}$$

Solution of the optimization problem that is the maximum geometric margin of S will be:

$$\gamma = \frac{1}{\|w'\|} \text{ ---------- (28)}$$

7.3.Support Vector

The Karush-Kuhn-Tucker complementary conditions look into the idea of Support Vector,

$$\alpha_i'[y_i(\langle w'.x_i\rangle + b') - 1] = 0, i = 1 \cdots l \text{ ---------- (29)}$$

This equation says that if functional margin of a data point is not 1, it will have its Lagrange multiplier α_i' as zero or will not contribute to the evaluation of optimal hyper-plane. Only the points with unit functional margin contribute to the calculation of optimal hyper-plane and called as *Support Vectors*. This is why maximal margin hyper-plane algorithm is called Support Vector Machine. However, it is the simplest one to study and there are other advanced SVM also available.

8. APPLICATIONS OF SVM IN INFORMATION PROCESSING

8.1.Overview

This chapter is completely focused on the core idea of current thesis work that is how we can leverage all the basics ideas and mathematics that we have developed so far. As far as algorithm is concerned, there has been practical implementation of SVM and software can be downloaded from internet. Among the most frequent ones are SVM [LIB] and SVM-Fu. In any real life application of SVM, it will be a wise idea to use any of the existing algorithm implementation as a core engine and then develop a wrapper on it to focus on application itself. This chapter talks about approaches, challenges, issues and variations in various possible applications of SVM. The discussion is in quite general sense. However, later one of the application will be picked up as a specific case and will be seen in complete details. That will also be the place where we will also delve into details of the core engine of SVM.

8.2.Challenges & Variations

In entire mathematical formulation we started with some training data and idea is to train the machine and then being able to test it for new and unseen test data. As discussed above, typically these data points are the vectors of attributes. One of the challenges that are the first one to be faced before SVM can be used efficiently is how we choose attributes. The selection of attributes affects results drastically. If data can not represent the problem, SVM is of no use at all. As an example, in Face Recognition problem, one needs to represent image of a face completely. A face can be represented based on large number of attributes like color of eyes, distance between two eyes, distance between two ears, length of nose or may be color of hairs. There can be just too many attributes to be considered. It will be safest idea to include

among all of them but some of the features might be easy to be extracted, others might be quite tedious. So there is a trade off between number of attributes that one chooses to pick up and time and effort required to generate data set.

As seen above, hyper-plane might be next to the impossible to be spotted in input space so there is then this idea of kernel functions. Depending on how one sees feature space, the space of classification function may vary. The selection of proper kernel function is an art to an extent and requires a closer understanding with the data and problem itself and at times might be tedious in terms of choosing right mathematical function.

The next decision that one has to take is what kind of SVM is to be used. Current thesis work has visited only Maximal Margin Hyper-Plane based SVM but in reality different kind of SVM exist just because of the fact that they decide to choose some other optimization criteria. Sometimes application can demand a different optimization criteria and one may be left with no option other than formulating SVM for that criteria. Here is an example from on of the research paper (ANGHELESCU & MUCHNIK) that has looked at a different SVM solely because it was the demand of application:

"A common context for designing text classifiers consists of problems for which the training data contains a small number of positive examples and a much larger number of negative ones. For instance, for information retrieval algorithm evaluation purposes in the environment of TREC, a standard test data was created using the Reuters RCV1 corpus and defining 100 different topics. For each such topic, there are

available some 10-400 positive examples and some 100-1500 negative ones. This example is not singular; other text databases show similar characteristics.

Such common features of the training data, namely very unbalanced proportions of positive/negative examples, pose difficult questions to many machine learning algorithms and they often lead to building poor classifiers that label all documents as negative.

We used a 1-nearest neighbor classification algorithm on the aforementioned 100 TREC topics, in the original term space (that is, the union of terms in all labeled training documents for a given topic), and in almost all cases this resulted in classifying all documents as negative. Moreover, support vector machines, which, at least theoretically, are the most powerful classification methods, also gave bad results. Based on an analysis of local observations around the error points obtained from these experiments, we devised the hypothesis that the main cause of the bad results was the aforementioned bias in data."

One of the factors that one needs to take care in this context is the dimensionality of data and the level of noise present in the data. In the formulation discussed in current thesis work, noise has not been accounted at all but there are variations of SVM that accounts for noise.

Additionally, binary classification is only one aspect of SVM. There are other uses also like Multi-classification and Regression that require a modified mathematical formulation of what has been discussed here.

In order to get a better feel of these variations, let us have a look at different real world applications of SVM.

8.3.Image Recognition/Classification

With the explosion of web page development, the availability of color scanners, printers, and digital media, people now have access to hundreds of thousands of images Since images have always been a popular means of communication, internet is going to exploit it to build increasingly large image databases. Google has already come up with a altogether different functionality of being able to search an image over internet. Searching an image will be a lot easier if we can categorize them. Once we have some framework of categorization set, any new image can be tested against this framework and included at right location. This will make image database much easier to maintain and use.

Also images have been integral part of research work; many researchers are cautious about distributing their work in fear that it may be copied illegally or represented as another's work. If this occurs then how can one make sure that an image is the one that he\she has developed and is his or her own property. Additionally, these images may be distorted from the original, yet we want to identify it as being a descendant of some original image. We need some way to recognize an image.

SVM can be used as a powerful tool to recognize image or to categorize them in some pre-defined classes. An image can be processed to generate a vector of attributes. The number of attributes needed to give a practical representation of an image can be substantially higher. Unlike conventional classification approaches, SVM does not care the dimensionality of data and is the best bet to be used. A crude approach that does not require any complex image processing is to see an image as a bitmap and form a one dimensional vector where every attribute gives pixel color at some location of bitmap. A color image gives rise to 3 types of basic colors (RGB) for a

given pixel so size of vector becomes 3 times the number of pixels that are being chosen to represent that image. In the case of grey level images number of attributes of a vector is the same as the number of pixels chosen to generate that vector from some image.

An image recognition problem will become a binary classification problem where slight variation of same image is labeled as +1 and any significant variation is labeled as -1. Train the SVM and user is all set to recognize any new image to see whether it is similar to what it has been trained for or is quite different from it.

If it is the problem of classification, numbers of classification labels are first agreed upon. An image is then processed to generate a vector of attributes and then assigned appropriate label. Data is then trained based on multi-classification problem SVM and label of any new image can be tested against the trained SVM.

8.4.Function Approximation & Regression

Function approximation and regression problems seek to determine from pairs of examples (x, y) an approximation to an unknown function y=f(x). The application of SVM to such problems has been intensively benchmarked with "synthetic data" coming from known functions. Although this demonstrated that SVM is a very promising technique, this hardly qualifies as an application. There are only a few applications to real world problems. For example, in the Boston housing problem, house prices must be predicted from socio-economic and environmental factors, such as crime rate, nitric oxide concentration, distance to employment centers, and age of a property.

8.5.Protein Structure Prediction

Proteins consist of more than hundreds of amino residues. Each protein forms particular tertiary structures. The tertiary structure can be classified into three substructures, named secondary structures. Several systems predict these secondary structures by using neural networks based on the information of consecutive residues ranging from seven to twenty one.

Recently, the task of predicting protein structure from protein sequence has been seen an important application of support vector machines. A protein's function is closely related to its structure, which is difficult to determine experimentally. There are mainly two types of methods for predicting protein structure. The first type includes threading and comparative modeling, which relies on a priori knowledge on similarity among sequence and known structures. The second type, called ab-initio methods, predicts the protein structure from the sequence alone without relying on the similarity to known structures. Since SVM represents a new approach to supervised pattern classification which has been successfully applied to a wide range of pattern recognition problems, including object recognition, speaker identification, gene function prediction with micro array expression profile, etc. In these cases, the performance of SVM either matches or is significantly better than that of traditional machine learning approaches, including neural networks.

8.6.Spam Detection

Spam detection using Support Vector Machine is another interesting application that has attracted researchers recently. Spam can be seen as an email message that is unwanted or in other words is electronic version of junk mail that is delivered by the postal service. One of the reasons for the proliferation of spam is that bulk e-mail is

very cheap to send and although it is possible to build filters that reject e-mail if it is from a known spammer, it is easy to obtain alternative sending addresses.

Solutions to the proliferation of spam are either technical or regulatory. Technical solutions include filtering based on sender address or header content. The problem with filtering is that sometimes a valid message may be blocked. Thus, it is not our intent to automatically reject e-mail that is classified as spam. Rather, we envision the following scenario: in the training mode, users will mark their e-mail as either spam or non-spam. After a finite number of examples are collected, the learning machine will be trained and the performance on new examples predicted. The user can then invoke the e-mail classifier immediately or wait until the number of examples is enough such that performance is acceptable. After the training mode is complete, new e-mail will be classified as spam or non-spam. In one presentation mode, a set of new email messages is presented in a manner consistent with the time of delivery and the spam messages color-coded. It is then up to the user to either read the e-mail or trash the email.

An alternative presentation mode is to deliver e-mail to the user in decreasing order of probability that the e-mail is non-spam. That is, e-mail with high probability of being a spam.

It is highly desirable that if the user decides that e-mail messages be rank-ordered by degree of confidence that the rank ordering be reliable. By reliable, we mean that the user can either start at the top of the list of e-mail messages or be fairly confident that they represent non-spam messages or start at the bottom of the list and be confident that the messages are spam. It is only near the middle of the list (low confidence) that

it is reasonable to the user that a few non-spam or spam messages may be misclassified. Therefore, it is important that our learning algorithm not only classify the messages correctly but that a measure of confidence is associated with that classification so that the message can be rank ordered.

The feature identification for representation of a spam is quite subjective. The most popular ones are email of sender, the amount of text contained in an email, time at which it arrive and subject title of email. However one has to be very careful while deciding on the choice of attributes and need to make sure that they are easy to evaluate and best representatives of the spam.

One of the design choices is between using some of the features or all of the features. One possible advantage of using a finite number of features is better generalization. By generalization we mean that good performance on the training set generalizes to good performance on a separate test set. Depending on the kind of SVM (classification criteria and kernel mapping) it may be the case that there is an optimum set of features, less than the total number of available features. For example, if the dimensionality of the classification space is greater than the number of examples, then the examples may always be separable by a non unique hyper-plane with zero training error (assuming the patterns are independent). Since there are, in general, an infinite number of separating hyper-planes, one does not obtain the optimal separating hyper-plane (the one that has the best test performance).

8.7.Support Vector Decision Tree

A decision tree takes certain number of independent parameters and outputs yes or no output. Therefore decision tree is a binary tree having its node values as Boolean values. Every node is the result of a decision and will require certain number of

attributes. Since SVM can be used to train a system to arrive at a result based on given parameters. So SVM finds a crucial role in forming a decision tree. It means that SVM can be used to train every node of binary tree and whenever a decision has to be tested, parameters need to be evaluated and fed at different nodes of tree. Trained SVM will give values of various nodes and will form proper decision tree.

8.8.Text Categorization

The task of text categorization is the classification of text based documents in some pre-defined categories. Documents can be classified based on text categorization. In fact, classification of documents was the inspiration for the start of this thesis work. A web services based framework was developed as a part of M. Eng. Project at Dept. of Civil & Environmental Engineering at MIT in 2003-04. Framework was smart enough to automatically classify documents from a sharable folder and a network of users was built to share classified documents among them. However, framework lacked a proper content based classification algorithm and supported only a crude form of classification scheme that picks up 5 most frequent words after filtering out all regular words.

The problem of text categorization also arise in other applications like email filtering, web searching, office automation and sorting documents by their relevance. Since a given document can be classified into multiple categories so this is basically a Multi-class classification problem and will require slight modification of mathematical formulation seen in previous chapters.

One of the most common techniques to classify a document is Vector-Space Model. A document is seen as a vector of Boolean values. Every Boolean attribute corresponds to whether document belongs to some category or not. The numbers of

categories are fixed and need to be decided before one move into process of mapping document to its corresponding vector. In fact category identification for a document is basically process of searching some keyword in a document. This type of approach is easy to be automated as a document can be parsed by computer program and presence of keyword can be checked against it. In reality, one time occurrence of a keyword in a document does not necessarily indicate the fact that document corresponds to the category in consideration. So in order to make SVM based solution more effective, weight of a keyword is evaluated in a document rather than checking its one time presence in the document. Finally, the vectors are normalized to remove the information about the length of text in a document.

8.9.Handwriting Recognition

During the last years the task of electronic handwriting recognition has gained an immense importance in all-day applications, mainly due to the increasing popularity of the personal digital assistant (PDA). Currently a next generation of ``smart phones'' and tablet-style PCs, which also rely on handwriting input, is further targeting the consumer market. However, in the majority of these devices the handwriting input method is still not satisfying. Current PDA still use input methods abstracting from the natural writing style, e.g. in the widespread Graffiti.

Thus there is demand for a handwriting recognition system which is accurate, efficient and which can deal with the natural handwriting of a wide range of different writers. One of the key elements in handwriting is signature. The ability to recognize signature is essential for the success of online shopping through use of PDA. This is the application that has been implemented through the use of SVM in this thesis work. The Microsoft .NET framework and C# language has been used to build an

application that allow user to draw signatures and generate a test file. The application then utilizes SVM^{LIB} (software developed for the implementation of SVM algorithm) as a core engine to train system. Once SVM is trained up to satisfaction, testing platform provided by application can be used to test validity of any new signature. The next chapter digs the every single detail of this application and important results that convince us for the promising future of SVM.

9. SIGNATURE RECOGNITION USING SUPPORT VECTOR MACHINE

9.1.Overview

The *signature recognition* is the process of verifying the writer's identity by checking the signature against samples kept in a database. Despite of the fact that signature of the same person differ from time to time, there is an element of similarity shared among them. Current thesis work use SVM as a way to extract that common element and train the machine for a set of data. Once SVM is trained for a set of signatures that are more or less same (because they are drawn by same person), it can to recognize whether a new signature belongs to same person or not.

The key steps used in this implementation are signature representation, data capture, feature extraction and generating a test platform.

9.2.Signature Representation

Two basic representations of signatures are considered in this thesis work. One is based on the color of the pixel and second is based on the time series.

9.2.1. Pixel Based Approach

User is asked to draw signatures on a bitmap and as mouse touches the bitmap, a curve of sufficient thickness and with a predefined color is drawn on bitmap. This brings the different among the color of the pixels as most of them still have their original color and are untouched and rest of them has got their color changed. Based on the color of every pixel a vector of hue (RGB say R) is created and vector of R value is generated. Color of the signature is chosen in such a way that the pixel that is unaffected produce zero R value. Also first element of the vector is kept the classification category of that signature which is a binary number. In the current convention, the value of +1 implies that signature drawn in bitmap is

the right signature and the one with values of -1 are wrong ones. It is beneficial to include enough data of both the categories; otherwise trained SVM will be bias.

Normalization is required as the same signature when drawn in different proportions will lead to increase in the variety of data points. In order to achieve an increased level of accuracy, the original bitmap is mapped to relatively smaller one. It not only reduces the number of attributes of the vector but also allow user an opportunity to draw signature on the larger bitmap with a sufficient level of comfort.

In order to increase the accuracy of method, it is advisable to affect a significant portion of pixels. If numbers of affected pixels are pretty low, we will require tremendous training examples to train SVM for just one person's signature as most of the attributes in the vector will be zero. On the other hand if we increase thickness of signature very high, most of the pixels will get affected. As number of participating pixels increases combination of R-values increase so it again requires large set of data will be required to train SVM effectively. This is the

reason that optimum thickness of signature is required.

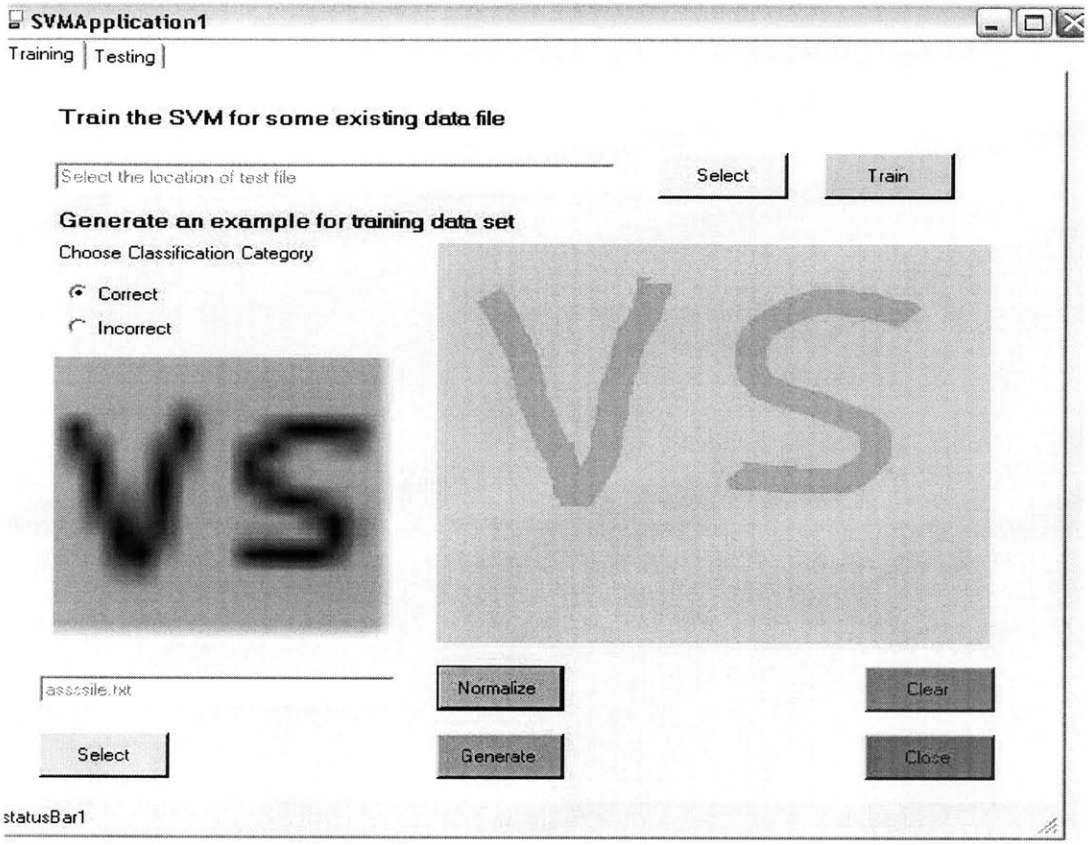

Figure 5: Pixel based Signature Recognition (Data Generation)

Figure 4 shows the platform of data generation. User draws a signature with mouse, selects the appropriate classification category and normalizes it. Every time a signature is normalized on bitmap, a vector of R-values is created and then fed to the SVM. User can specify the training data file and every signature just gets appended to it. Once SVM is sufficiently trained, machine is all set to test a new signature.

It is important to note that even while testing a new signature, same kind of vector has to be created to maintain consistency of new example with training data. This means user need to normalize every time he\she wants to test a new signature.

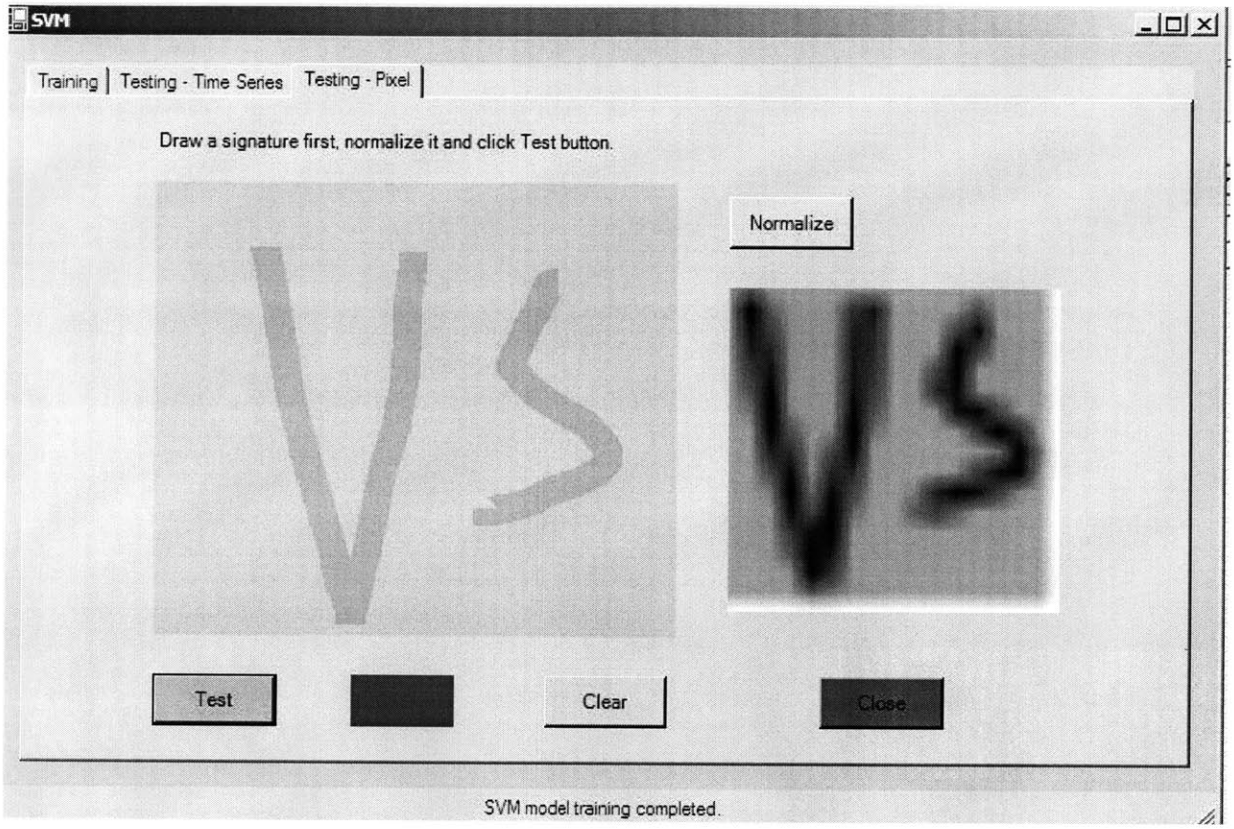

Figure 6: Correct Recognition of "VS"

The main disadvantage with this method is the uneasiness of drawing a signature with mouse. It is quite difficult to draw a set of similar signatures with a mouse on screen. In fact at times, drawing just one real signature on screen with mouse takes lots of attempt and becomes a painful process. This is the reason that only name initials are treated as signatures and tried with this application. Figure 4 shows training for name initials of "VS". Figure 5 shows correct identification of "VS" initials by the trained SVM. (See green light between Test and Clear button). Figure 6 shows that a properly trained SVM is smart enough to reject anything that does not look like a "VS". (See red light between Test and Clear button).

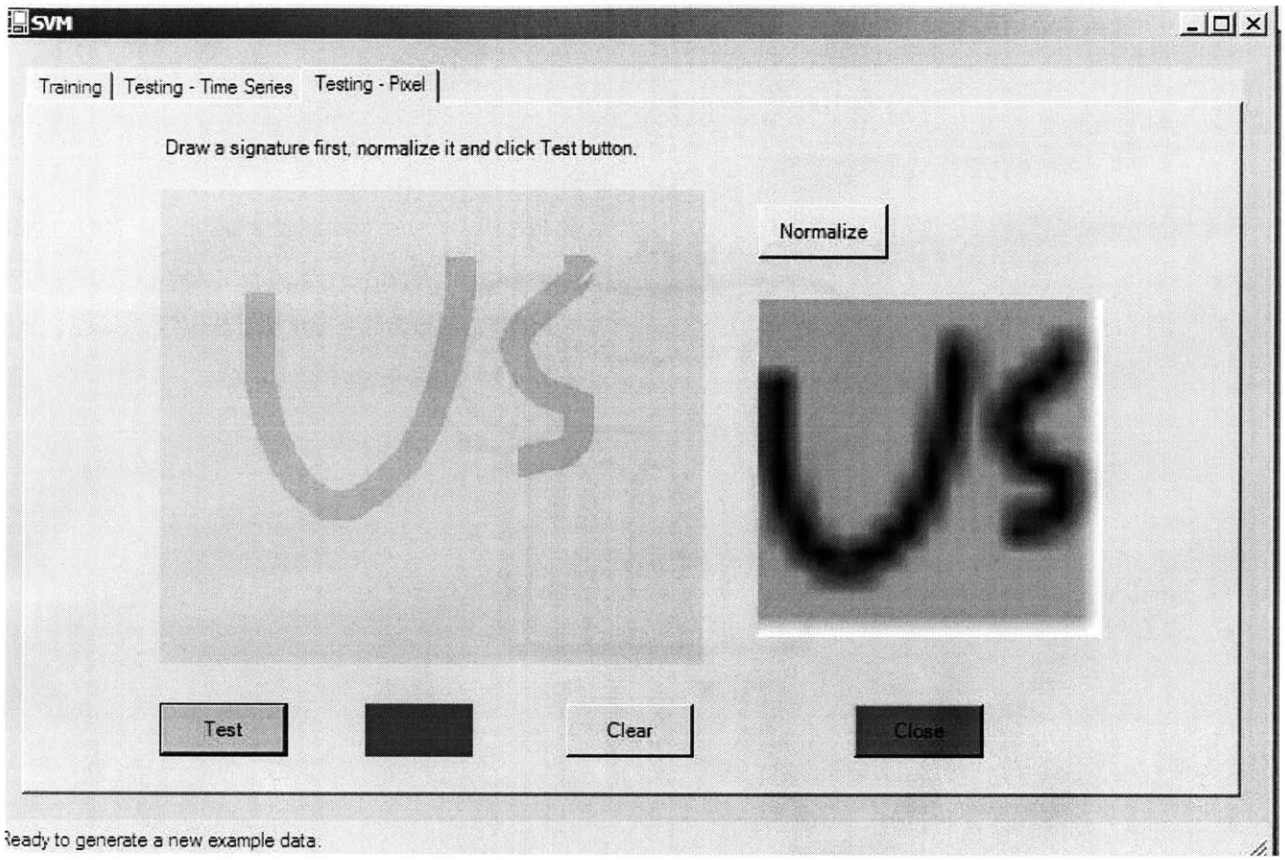

Figure 7: SVM denies signature to be "VS"

9.2.2. Time Series Based Approach

Generation of time series based on the movement of ink-pen is considered another way of representing signatures. Tablet PC is used to utilize features of ink pen and Tablet based APIs are explored to provide a great level of comfort to user. Not only it is easier for user to draw signature or write using ink pen but also it is effective for application to catch the movement of ink pen. The picture-ink control provided by Microsoft's Tablet PC API has been used for drawing of signatures.

Figure 7 shows that as user draw signature on picture-ink using ink pen, a time series of location of tip of pen (x coordinate followed by y coordinate) is generated. Figure 7 display time series in a list-box.

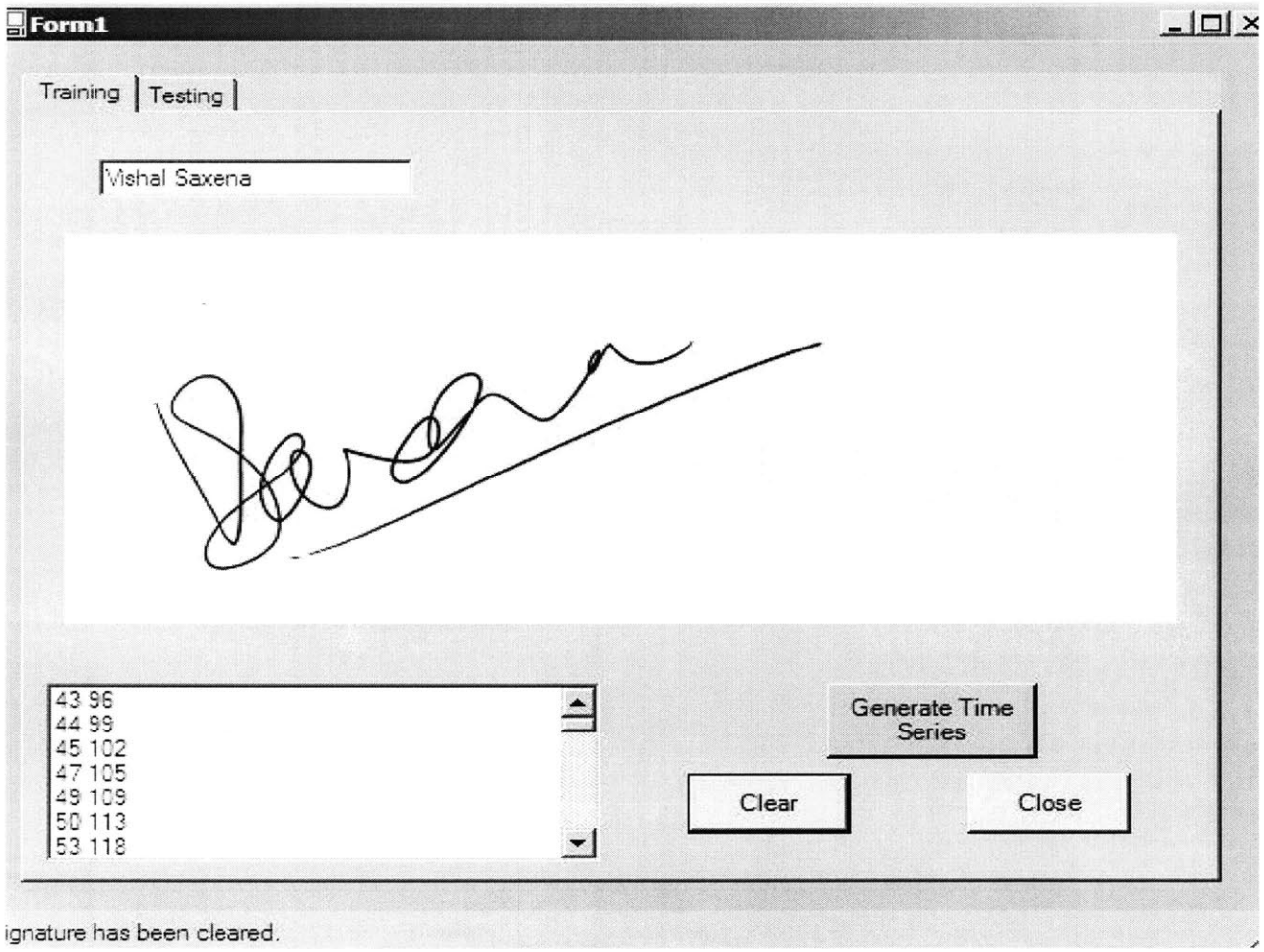

Figure 8: Generation of Time Series Data for Signature

Since this strategy believes in the generation of time series vector based on movement of ink-pen, time taken by user to draw a signature imposes some restrictions on the use of SVM. A user can draw same signature at different speed

and hence based on the duration of touch between ink pen and picture-ink control, the length of time series varies. Since current implementation of SVM core engine and even the mathematical formulation of SVM that has been studied here does not account for varying length of attribute vectors, it is essential to maintain some constant length of time series.

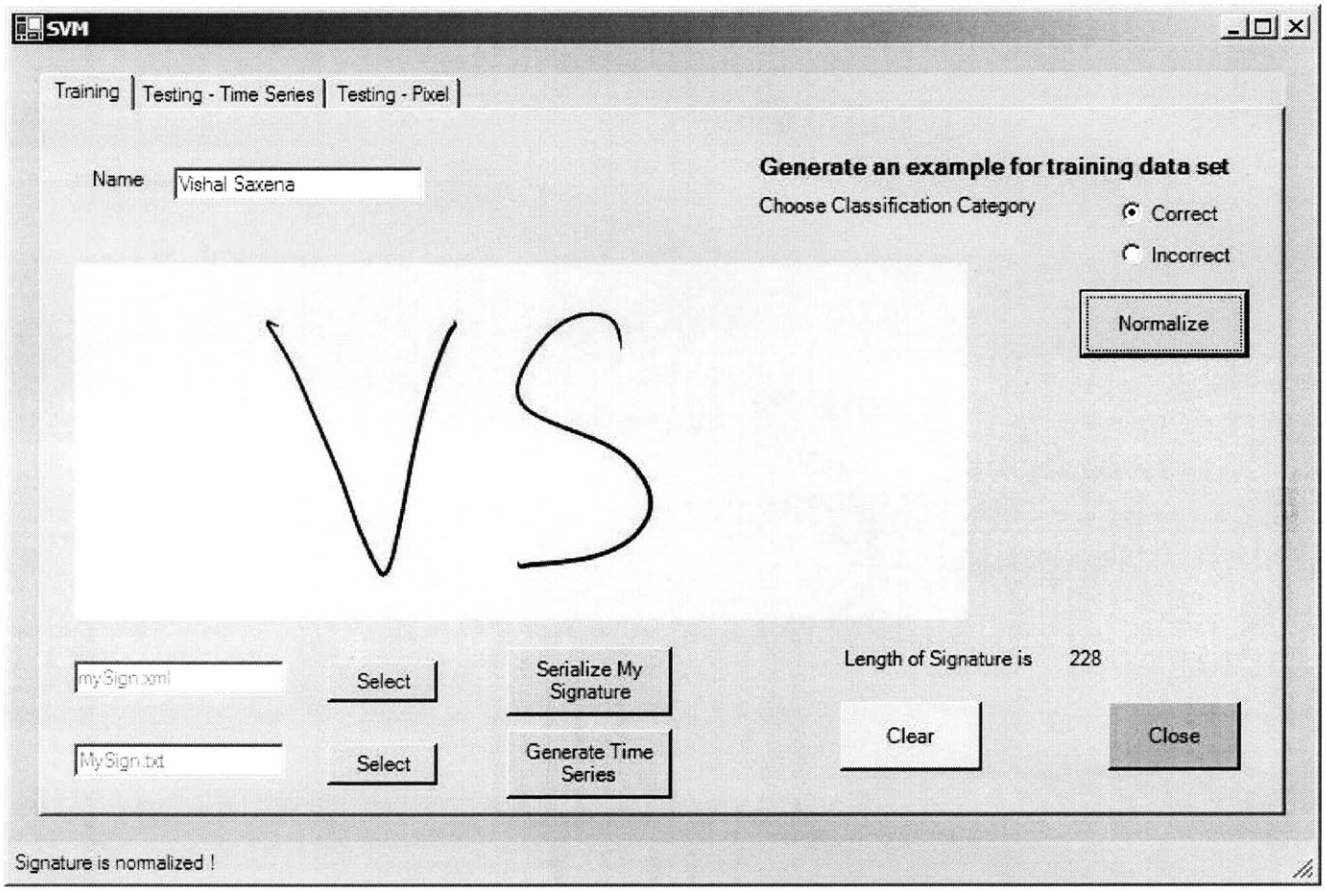

Figure 9: Time Series based Recognition

Figure 8 shows a name initial "VS" when drawn with some speed gives a 228 time unit length of time series vector (this means 114 points of touch have been recorded and every point gives two coordinate, X and Y. In order to capture signature of same length two strategies has been followed.

- Minimum length of signature is decided and put as a check in the program. This program will not add time series of a signature to given test file if its length is lesser than specified minimum length. It means user has to discard all those signatures that don't carry sufficient length of time series.
- If length of a signature is more than the minimum length, program just picks up the length up to specified minimum length and discards everything thereafter. It is left to the prudence of user to decide up to what length of signature he\she wants to be added to the test file. It has been studied 5% extra length of signature can be easily added and curtailment of last 5% length by program does not add any significant noise. However if any length above minimum is allowed to be added to test file, it just add lot of noise and hinders performance of trained SVM.

  Every time user draws a signature on test or training platform, it can be of different size because it is natural for user not being able to replicate everything of same shape and size. The tendency of not being able to draw same shape is fine with application as identification of similar shapes is the entire idea about signature recognition. But size should be kept uniform otherwise SVM will require tremendous amount of data depending upon the irregularities on the size of user generated signature. In fact test file generated is also platform size specific. Just to eliminate effect of the size of platform and variation in the size of signature by user, a normalization is being done before user add anything to test file or test any signature from test platform. Normalization is mandatory throughout the use of application otherwise data will not be consistent.

Figure 8 shows the training platform. User first draw a signature, chooses appropriate categories and click normalize button. This cause generation of time series, its normalization, addition of classification category to the start of time series vector and series getting appended to the given test file. User can browse up to the test file. The application also offers an opportunity to user to be able to create an XML just to see the data structure of signature supported by application.

A time series based SVM has been trained for testing of name initials "VS". Figure 9 shows correct identification of "VS" by trained SVM. On the other hand anything of different shape than that of "VS" will be clearly rejected by SVM as shown in figure 10. The green light and red light shown between the "Test" and "Clear" button shows correct and incorrect classification respectively.

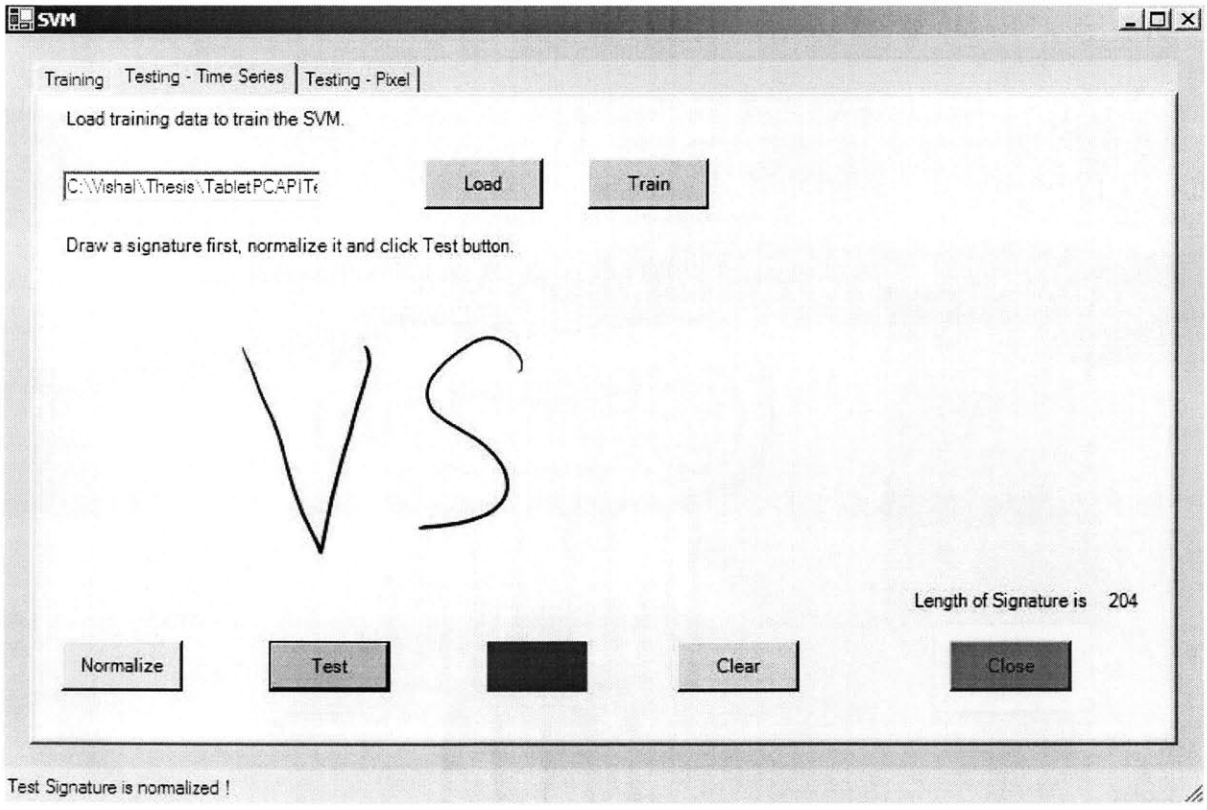

Figure 10: Identification of "VS" by Time Series SVM

Since it require lot more training data to train a SVM for time series based signature recognition as compared to pixel based signature, so simplest signature has been tried here. Name initials being the simplest kind of signature have been tested a lot with this application. Continuing in the same direction, recognition of real world signature would be the focus for this application.

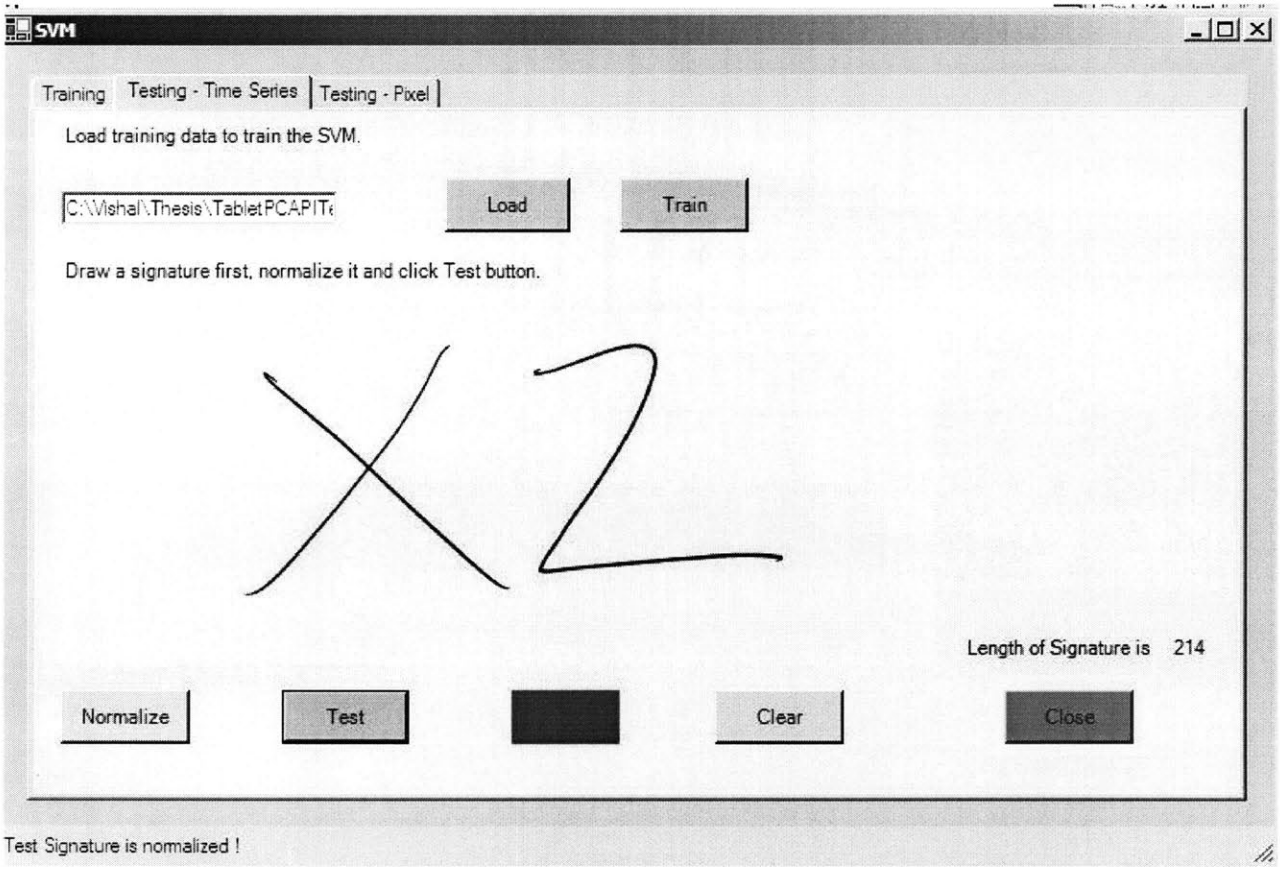

Figure 11: Time Series based SVM denies signature to be "VS"

Training of real world complicated signatures (Time Series Based Recognition) in strategy doesn't have any difference as compared to recognition of name initials.

The only difference is that number of samples needed to train SVM will be substantially higher.

Based on the results (see the appendix), around 30-40 signatures (including correct and incorrect) sample signatures are needed to train SVM for "two letter based name initials" for pixel based representation of signature approach. The numbers of training samples for almost same accuracy of recognition of similar name initials rise to 70-80 for time series based approach. However, this number is expected to rise up to anything more than 100 for classification of real world signatures (you just keep dragging pen to sign your real signatures!) based on time series representation.

10. CONCLUSION

The theory of SVM appears quite promising and offers numerous scientific and industry specific applications. The mathematical basis of the Support Vector Machine is huge and will remain attractive to find more efficient variations of new SVM. Current thesis does not focus on performance improvement of core optimization engine of SVM and achieving computing performance or optimizing memory need to be attraction of next level research.

The application of Support Vector Machine to recognize SVM has been seen in two different ways. As far as ability to recognize signature of similar shapes is concerned, Time series based signature representation is more efficient as compared to pixel based approach. However time series approach demands more examples data to be able to train SVM up to the same expectation as it is required by its pixel based approach.

Tablet PC has given more dimensions to the solution of signature recognition as user is now able to replicate what he\she used to draw on paper. This way, contribution of Tablet PC to the ability of SVM to solve the problem of handwriting recognition can be appreciated in real world applications. In future, it will be much easier for user to validate documents on PDA or writing and signing documents through some online tool.

Microsoft .NET support and its Tablet PC based API (C#) gives developer an added advantage over other traditional programming languages like C, C# and Java to be able to develop a user interface where user is able to test signature and provides efficient way of backend computer processing to generate test file and load them to train and test the applicability of SVM

Finally, results obtained in this thesis work are quite satisfactory. The pixel based application is quite correct in its implementation as it does not involve any assumption is

test data generation, training or even in testing. The only subjectivity that can affect the results is the width of signature inbuilt in application that is used to affect the pixel colors. One can aim a separate research to arrive at the optimum width of signature drawn to achieve the training using minimum number of test examples. Accuracy achieve in the results (the number of successful test out of total test) is also subjective to comment. Around 30-40 signatures have been used to recognize signatures in its pixel based representation and accuracy achieved is up to 90%. However, the kind of signature samples that can be generated and hence trained and tested here is very limited just because of the fact that mouse is being used to write signatures.

On the other hand Time Series based application attempt to capture the minute details of signatures and due to use of Tablet PC is the best suited for the real world signatures. Given the complex nature of signature drawing, one need to generate large data samples and need lot of time to carefully generate set of similar real signatures. The restrictions forced like minimum length of signature in Time Series based application, further demand more effort as user need to make sure that signatures attain a minimum length. Given these complications in Time Series Based Application, there is a need to make this application more users friendly. At t the same time, one needs to revisit design of time series generation so that signatures of any time length can be trained and tested by SVM.

REFERENCES

- "An introduction to Support Vector Machines and other kernel-based learning methods" by Nello Cristianini & John Shawe-Taylor. (http://www.support-vector.net/)
- A. Bell and T. Sejnowski. An information-maximization approach to blind source seperation and blind deconvolution. Neural Computation, 7:1129--1159, 1995.
- B. V. Dasarathy. Nearest Neighbor Norms: NN Patern Classifaction Techniques. IEEE Computer Society Press, 1991.
- Christopher Chatfield and Alexander J. Collins. Introduction to Multivariate Analysis. Chapman and Hall, 1980.
- David E. Rumelhart, James L. McClelland, and the PDP research Group. Parallel distributed processing: explorations in the microstructure of cognition. MIT Press, 1986.
- Ellen M. Voorhees and Donna Harman. Overview of TREC 2001. In Proceedings of the 2002 Text Retrieval Conference, 2001.
- "Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning" a PhD dissertation submitted by Ryan Michael Rifkin to the Sloan School of management science. (http://www.ai.mit.edu/projects/cbcl/publications/theses/thesis-rifkin.pdf).
- "Face Recognition Using Component-Based SVM Classification and Morph-able Models" by Jennifer Huang & Bernd Heisele at Artificial Intelligence Laboratory, MIT. (http://www.csail.mit.edu/research/abstracts/abstracts03/computer-vision/11huang.pdf)

- John R. Williams. 2003. Proposal to Shell: Agent Based Computing and Agents for Data Processing. IESL, MIT.
- J. R. Quinlan. C4.5: programs for machine learning. Morgan Kaufmann, 1992.
- Judea Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, 1988
- LIBSVM - A Library for Support Vector Machines By Chih-Chung Chang and Chih-Jen Lin, National Taiwan University. (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
- L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4, 1996.
- Microsoft Visual C# http://msdn.microsoft.com/vcsharp/.
- Microsoft Visual J# http://msdn.microsoft.com/vjsharp/.
- "Optimization of SVM in a space of two parameters: Weak margin and Intercept. Applications in text classifier design" by ANDREI V. ANGHELESCU AND ILYA B. MUCHNIK
- Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. In Proceedings of the Fifth International Workshop on Machine Learning. Morgan Kaufmann, 1988.

- Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. In Proceedings of the Fifth International Workshop on Machine Learning. Morgan Kaufmann, 1988.

- Reuters corpus 1996-08-20 to 1997-08-19
- Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press/Bradford Books, Cambridge, MA, 1998.

- SVM Application list available at world wide web http://www.clopinet.com/isabelle/Projects/SVM/applist.html
- SVM Light, An implementation of SVM in C. (http://svmlight.joachims.org/)
- Statistical Learning Theory and Applications, Course taught at MIT, Spring 2002 (http://www.csie.ntu.edu.tw/~cjlin/libsvm/).
- Vapnik, Vladimir (1998): Statistical Learning Theory (http://kiew.cs.uni-dortmund.de:8001/mlnet/instances/81d91eaae1752fe0be).

APPENDIX A

The perceptron algorithm implementation source code:

```
using System;
using System.Collections;
namespace Perceptron_Algorithm
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>
	class PrimalDual
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			//
			// TODO: Add code to start application here
			//
			string input;
			double neata = 1.0;	//default learning rate
			//input
			Console.WriteLine("Welcome to the The Primal Form
implementation for Perceptron Algorithm for Supervised Learning\n");
			Console.WriteLine("Description : Given a linearly seperable
training set S and learning parameter neeta , we can find a hyperplane sepearting set into
two categories \n");
			Console.WriteLine("Input: Please Enter the dimension of the input
space\n");
			input = Console.ReadLine();
			int dimension_Space = Int32.Parse(input);
			Console.WriteLine("Input: Please Enter the number of Training
Data Sample\n");
			input = Console.ReadLine();
			int no_Training_Sample = Int32.Parse(input);
			ArrayList inputList = new ArrayList();
			int[] Y = new int[no_Training_Sample];
			for(int i=0;i<no_Training_Sample;i++)
			{
				double[] X = new double[dimension_Space];
				Console.WriteLine("Input: Please Enter the Input Vector
X" + i + " \n");
				for(int j=0; j<dimension_Space;j++)
				{
```

```
                    Console.WriteLine("Input: Please Enter the
Element#" + j + " \n");
                    input = Console.ReadLine();
                    X[j] =  Double.Parse(input);
                }
                inputList.Add(X);
                Console.WriteLine("Classification Category +1 or -1?,
Enter only +1 or -1");
                input = Console.ReadLine();
                Y[i] = Int32.Parse(input);
            }
            Console.WriteLine("Please Enter the Learning Rate \n");
            input = Console.ReadLine();
            neata = Double.Parse(input);
            double[] W = new double[dimension_Space+1];      //Needed in
Primal Form
            double[] alpha = new double[no_Training_Sample+1];
    //Needed in Dual Form
            //Initialization
            for(int i=0;i<=dimension_Space;i++)
            {
                W[i] = 0;
            }
            for(int i=0;i<=no_Training_Sample;i++)
            {
                alpha[i] = 0;
            }
            double b = 0;
            int no_mistakes = 0;
            Console.WriteLine("Solution using Perceptron in its Primal Form
\n");
            //Running the main engine
            no_mistakes =
PrimalDual.runAlgorithmInPrimalForm(inputList,Y,W,b,neata,no_Training_Sample,dim
ension_Space);
            b = W[dimension_Space];
            Console.WriteLine("After correcting " + no_mistakes +" mistakes,
Classification has been arrived as \n:");
            for(int i=0;i<dimension_Space;i++)
            {
                Console.WriteLine("W[" + i + "] = " + W[i]);
            }
            Console.WriteLine("b = " + b);
            Console.WriteLine("Copyright @2004 Vishal Saxena, Please
contact at vsaxena@mit.edu for any feedback or comment \n");
```

```
        Console.WriteLine("*********************************************");
                        Console.WriteLine("Solution using Perceptron in its Dual Form
\n");
                        b = 0 ;

        PrimalDual.runAlgorithmInDualForm(inputList,Y,alpha,b,neata,no_Training_Sa
mple,dimension_Space);
                        b = alpha[no_Training_Sample];
                        double[] alphaDesh = new double[dimension_Space];
        //Needed in Dual Form
                        //Initialization
                        for(int i=0;i<dimension_Space;i++)
                        {
                                alphaDesh[i] = 0;
                        }
                        for(int j=0;j<dimension_Space;j++)
                        {
                                for(int i=0;i<no_Training_Sample;i++)
                                {
                                        double[] xj = (double[])inputList[i];
                                        alphaDesh[j] += alpha[i]*Y[i]*xj[j];

                                }
                        }
                        for(int i=0;i<no_Training_Sample;i++)
                        {
                                Console.WriteLine("Alpha[" + i + "] = " + alpha[i]);
                        }
                        for(int i=0;i<dimension_Space;i++)
                        {
                                Console.WriteLine("alphaDesh[" + i + "] = " +
alphaDesh[i]);
                        }
                        Console.WriteLine("b = " + b);
                        Console.WriteLine("Please enter for Exit\n");
                        input = Console.ReadLine();
                }

                //Page 12 : Chapter 2: Linear Learning Machines : SVM By Cristianini &
Taylor
                private static int runAlgorithmInPrimalForm(ArrayList X,int[] Y,double[]
W,double b, double neeta,int l,int d)
                {
                        int k = 0;
                        double R = 0;
```

```
                //Calculate R as the maximum of the Norm for all of the training
samples
                for(int i=0;i<l;i++)
                {
                        double[] xi = (double[])X[i];
                        double sum = 0;
                        for(int j=0;j<d;j++)
                        {
                                sum += Math.Pow(xi[j],d);
                        }
                        double temp = Math.Pow(sum,1.0/d);
                        if(R<temp)
                        {
                                R = temp;
                        }
                }

                //Now revising W and b based on the mistake made
                for(int i=0;i<l;i++)
                {
                        double[] xi = (double[])X[i];
                        double sum = 0;
                        for(int j=0;j<d;j++)
                        {
                                sum += xi[j]*W[j];
                        }
                        sum = sum *Y[i] + b;
                        if(sum <=0)
                        {
                                for(int j=0;j<d;j++)
                                {
                                        W[j] += neeta*Y[i]* xi[j];
                                }
                                b += neeta*Y[i]*R*R;
                                k = k+1;
                        }
                }
                W[d] = b;
                return k;
        }


        //Page 18 : Chapter 2: Linear Learning Machines : SVM By Cristianini &
Taylor
        private static void runAlgorithmInDualForm(ArrayList X,int[] Y,double[]
alpha,double b, double neeta,int l,int d)
```

```
{
        double R = 0;
        //Calculate R as the maximum of the Norm for all of the training
samples
        for(int i=0;i<l;i++)
        {
                double[] xi = (double[])X[i];
                double sum = 0;
                for(int j=0;j<d;j++)
                {
                        sum += Math.Pow(xi[j],d);
                }
                double temp = Math.Pow(sum,1.0/d);
                if(R<temp)
                {
                        R = temp;
                }
        }

        //Now revising Alpha and b based on the dual check
        for(int i=0;i<l;i++)
        {
                double[] xi = (double[])X[i];
                double sum = 0;
                for(int p=0;p<l;p++)
                {
                        double[] xp = (double[])X[p];
                        for(int j=0;j<d;j++)
                        {
                                sum += xi[j]*xp[j];
                        }
                        sum = sum *Y[p]*alpha[p] + b;
                }
                if(sum <=0)
                {
                        for(int j=0;j<l;j++)
                        {
                                alpha[j] = alpha[j] + 1;
                        }
                        b += Y[i]*R*R;
                }
        }
        //storing b as the last element in Alpha Vector
        alpha[l] = b;
}
```

```
        }
}
```

Test Run – Here is an output for one of the test run of above mentioned source code for Perrceptron Algorithm.

This is the output from console based application for some given set of two dimensional training examples.
The numbers of test examples are 10 and dimension of space is also 2.

```
Welcome to the Primal Form implementation for Perceptron Algorithm for Supervised Learning
Description: Given a linearly separable training set S and learning parameter neeta , we can find a hyper plane separating set into two categories
Input: Please enter the dimension of the input space
2
Input: Please enter the number of Training Data Sample
10
Input: Please Enter the Input Vector X0
Input: Please Enter the Element#0
12
Input: Please Enter the Element#1
23
Classification Category +1 or -1?, Enter only +1 or -1
1
Input: Please Enter the Input Vector X1
Input: Please Enter the Element#0
34
Input: Please Enter the Element#1
56
Classification Category +1 or -1?, Enter only +1 or -1
1
Input: Please Enter the Input Vector X2
Input: Please Enter the Element#0
100
Input: Please Enter the Element#1
120
Classification Category +1 or -1?, Enter only +1 or -1
1
Input: Please Enter the Input Vector X3
Input: Please Enter the Element#0
-54
Input: Please Enter the Element#1
23
Classification Category +1 or -1?, Enter only +1 or -1
-1
Input: Please Enter the Input Vector X4
```

Input: Please Enter the Element#0
12
Input: Please Enter the Element#1
-34
Classification Category +1 or -1?, Enter only +1 or -1
-1
Input: Please Enter the Input Vector X5
Input: Please Enter the Element#0
-24
Input: Please Enter the Element#1
-45
Classification Category +1 or -1?, Enter only +1 or -1
-1
Input: Please Enter the Input Vector X6
Input: Please Enter the Element#0
123
Input: Please Enter the Element#1
213
Classification Category +1 or -1?, Enter only +1 or -1
1
Input: Please Enter the Input Vector X7
Input: Please Enter the Element#0
23
Input: Please Enter the Element#1
34
Classification Category +1 or -1?, Enter only +1 or -1
1
Input: Please Enter the Input Vector X8
Input: Please Enter the Element#0
-54
Input: Please Enter the Element#1
-34
Classification Category +1 or -1?, Enter only +1 or -1
-1
Input: Please Enter the Input Vector X9
Input: Please Enter the Element#0
-23
Input: Please Enter the Element#1
-76
Classification Category +1 or -1?, Enter only +1 or -1
-1
Please Enter the Learning Rate
1
Solution using Perceptron in its Primal Form
After correcting 1 mistake, Classification has been arrived as
:

W [0] = 12
W [1] = 23
b = 60498

****
Please enter for Exit

APPENDIX B

The source code for the implementation of pixel based signature recognition has been provided in the CD attached with this thesis work. SVM LIB is being used as a core SVM engine for optimization of maximal hyper-plane. A dynamic link library can be downloaded from the web for SVM Lib and need to be added as a reference to the .NET project. In fact there is another reference need to be added in the project for "vjslib". The "vjslib" provides a communication between Java written SVM lib and .NET based classes that have been developed to provide platform for data generation, training and testing. Here is the one of the example of training data file that has been generated by this application and can be used to recognize "VS" name initials through its pixel based representation. Entire training file can be seen in the CD attached with this thesis work (myTestFile_VS.txt).

1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.992156862745098 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.992156862745098 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.490196078431373 -0.223529411764706 -0.686274509803922 -0.992156862745098 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.0274509803921568 1 1 1 -0.992156862745098 -1 0.780392156862745 -0.0823529411764705 -1 -1 -1 -1 -0.898039215686274 -0.843137254901961 -0.968627450980392 -1 0.662745098039216 0.694117647058824 -0.662745098039216 -0.67843137254902 -0.992156862745098 -1 0.388235294117647 0.529411764705882 -1 -1 -1 -1 0.411764705882353 0.945098039215686 -0.819607843137255 -1 0.835294117647059 0.254901960784314 -1 -1 -0.992156862745098 -1 -0.286274509803922 1 -0.592156862745098 -1 -1 -0.733333333333333 1 0.2 -1 -1 0.0352941176470588 1 -0.537254901960784 -1 -0.992156862745098 -1 -0.968627450980392 0.850980392156863 0.0901960784313726 -1 -1 -0.0745098039215686 0.929411764705882 -0.890196078431373 -1 -1 -0.945098039215686 0.780392156862745 0.647058823529412 -0.317647058823529 -0.992156862745098 -1 -1 0.0431372549019607 0.929411764705882 -0.615686274509804 -1 0.592156862745098 0.341176470588235 -1 -1 -1 -1 -0.490196078431373 0.92156862745098 0.388235294117647 -0.992156862745098 -1 -1 -0.945098039215686 0.709803921568627 0.435294117647059 -0.56078431372549 1 -

0.356862745098039 -1 -1 -1 -1 -1 0.52156862745098 0.270588235294118 -0.992156862745098 -1 -1 -1 -0.23921568627451 1 0.568627450980392 0.662745098039216 -0.984313725490196 -1 -1 -1 -1 -1 0.686274509803922 0.176470588235294 -0.992156862745098 -1 -1 -1 -0.874509803921569 1 1 -0.12156862745098 -1 -1 -1 -0.874509803921569 -0.607843137254902 0.16078431372549 1 -0.129411764705882 -0.992156862745098 -1 -1 -1 -1 -0.0823529411764705 0.16078431372549 -0.898039215686274 -1 -1 -1 0.435294117647059 1 0.92156862745098 0.0117647058823529 -0.984313725490196 -0.992156862745098 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.52156862745098 -0.372549019607843 -0.937254901960784 -1 -1 -0.992156862745098 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098 -0.992156862745098

Note that first element of this vector is 1, it implies that this is the vector generated for correct (classification category = +1) training example. Basically user has drawn "VS" on board and generated the test data.

APPENDIX C

Once again, entire .NET project including training file, SVM Lib libraries used and C# source code (windows form) developed for the time series based recognition of signatures can be found out in the CD attached with this thesis work. Here is the one of the test examples that has been generated by this application and is the part of training file (VS.txt) for recognition of "VS". The minimum length of signature time series is kept at 200.

1 .0000 .0417 .0092 .0667 .0138 .0917 .0229 .1250 .0275 .1583 .0367 .1917 .0459 .2250 .0550 .2667 .0688 .3000 .0780 .3417 .0872 .3750 .0963 .4083 .1009 .4417 .1101 .4750 .1193 .5083 .1284 .5417 .1330 .5667 .1422 .5917 .1514 .6250 .1606 .6500 .1697 .6750 .1789 .7083 .1881 .7333 .1972 .7583 .2064 .7833 .2110 .8083 .2202 .8333 .2385 .8750 .2523 .9167 .3624 .9583 .3716 .9333 .3761 .9000 .3853 .8667 .3945 .8250 .4037 .7833 .4128 .7417 .4220 .7000 .4358 .6500 .4495 .6083 .4587 .5583 .4725 .5083 .4862 .4583 .5000 .4167 .5138 .3667 .5275 .3167 .5413 .2667 .5505 .2250 .5642 .1833 .5734 .1417 .5872 .1167 .5963 .0833 .9404 .0083 .9266 .0000 .9128 .0000 .8991 .0000 .8807 .0000 .8624 .0083 .8440 .0167 .8257 .0333 .8073 .0500 .7890 .0750 .7706 .1000 .7569 .1333 .7431 .1667 .7294 .1917 .7202 .2250 .7110 .2500 .7064 .2833 .7018 .3083 .6972 .3333 .7018 .3583 .7018 .3833 .7156 .4250 .7339 .4667 .7477 .4833 .7615 .5000 .7752 .5083 .7936 .5250 .8073 .5417 .8257 .5500 .8440 .5667 .8624 .5833 .8807 .6000 .8991 .6167 .9174 .6333 .9358 .6500 .9495 .6667 .9633 .6833 1.0000 .7917 .9771 .9000 .9450 .9417 .9312 .9500 .9174 .9583 .9037 .9667 .8853 .9750 .8670 .9833 .8532 .9833 .8349 .9917 .8211 .9917 .8073 1.0000

Note that first element of this vector is 1, it implies that this is the vector generated for correct (classification category = +1) training example. Basically user has drawn "VS" on board and generated the test data. The length of this vector is the length of time series i.e. 200.