The Support Vector Machined Kernel: Towards a New Classification Framework

By

Khaled Saeed Saad Zaghloul Aly Refaat

A Thesis Submitted to the

Faculty of Engineering at Cairo University

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

FACULTY OF ENGINEERING CAIRO UNIVERSITY

GIZA, EGYPT

May 2010

The Support Vector Machined Kernel: Towards a New Classification Framework

By

Khaled Saeed Saad Zaghloul Aly Refaat

A Thesis Submitted to the

Faculty of Engineering at Cairo University

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

Under the Supervision of

Prof. Dr. Amir F. Atiya

Cairo University

FACULTY OF ENGINEERING CAIRO UNIVERSITY

GIZA, EGYPT

May 2010

The Support Vector Machined Kernel: Towards a New Classification Framework

By

Khaled Saeed Saad Zaghloul Aly Refaat

A Thesis Submitted to the
Faculty of Engineering at Cairo University
In Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
COMPUTER ENGINEERING

Approved by the

Examining Committee

_____

Prof. Dr. Amir F. Atiya , Main Advisor

_____

Prof. Dr. Nevin M. Darwish , Member

_____

Prof. Dr. Aly Aly Fahmy , Member

_____

FACULTY OF ENGINEERING CAIRO UNIVERSITY

GIZA, EGYPT

May 2010

Acknowledgement

First and foremost, I am greatly indebted to the supervisor of this work, Prof. Amir Atiya, for his continuous support. We would like to acknowledge Prof. Nevin Darwish, and Prof. Magda Fayek (Cairo University); and Dr Pierre-Emmanuel Hladik, and Prof. Patrick Senac (LAAS CNRS) for the useful comments they provided. We specially thank Eng. Wael Nabil for preparing smart animations that have helped a lot in illustrating our proposed idea. Last, but not least, I would like to thank my mother for her patience and endless encouragement.

Abstract

In this thesis, we propose the so-called "SVM'ed-kernel function" and its use in SVM classification problems. This kernel function is itself a support vector machine classifier that is learned statistically from data. We show that the new kernel manages to change the classical methodology of defining a feature vector for each pattern. One will only need to define features representing the similarity between two patterns allowing many details to be captured in a concise way. The new proposed kernel shows very promising results. It opens the door for new feature definitions that could be created in various machine learning problems where similarity between patterns can be formulated more suitably.

Contents

List of Figures

List of Tables

Chapter 1

Thesis Overview

1.1 Motivation and Problem Definition

In classification problems, each training or test pattern is usually converted to a set of feature values forming a feature vector. Such value features could be of any type (numeric, categorical, periodic, or ordinal). These feature values are either used for model estimation (training), or for predicting the output corresponding to the input feature vector. As it was stated in the learning process steps, the stage of such conversion is essential for an ultimate success of the whole process.

1.1.1 Representation

Representing a pattern by a feature vector could be problematic in many real-world problems. In the following subsections, we introduce a number of classification problems, and show how representing a pattern by a feature vector could be problematic and discuss different alternatives.

1.1.2 Parrot Classification

If we want to represent a picture of a parrot to a feature vector in order to classify it into one of several parrot types. The easiest way is to consider the pixels of the image

as the used features. This will lead to about 1024×1024 features; learning with such huge number of features is very difficult due to the curse of dimensionality.

Another option is to define a set of a few effective number of features. The domain expert might propose that the color pattern of the parrot's head is important in differentiating between different classes. In this case, we will be faced by a serious problem which is how to represent such feature in one of the variable types. We might need to know all color patterns that are available and encode them in a feature definition.

Unfortunately, while knowing all color patterns within the training set is possible, knowing the real number needs a domain expert. If the number of color patterns is 1000, we could , for example, need 10 binary features.

Additionally, if such number of patterns is infinite due to new brands being artificially hybridized, encoding a pattern as a number of binary features will be impossible. We may then need to consider the large number of pixels representing the head as our numeric features which will affect model estimation negatively. Other option is to use a numeric variable to represent the infinite number of patterns. But, Assigning a value to such variable will be problematic and will lead to erroneous distance relationships being taken in consideration in the model estimation step. It will be very hard to represent a color pattern by a numeric variable and ensures that the distance between two values of such variable is a real distance measure. On the contrary, such distance will affect the model estimation step negatively since it will lead to close distances between different patterns and far ones between similar patterns.

If we are to define a feature vector representing similarity between two input parrot images, we will be able to define simple and effective high level features. One could for example define the one of the two following features to represent the color pattern of the head:

1) a categorical variable of two values (0 or 1). Such variable is assigned the value of 1 if the two parrots have identical head color patterns and 0 otherwise. 2) a numeric variable representing how similar the two color patterns of the two heads.

In this case, the first option shows simplicity since we have eliminated the number of categories from being very large or infinite into only two categories (identical-not

Figure 1.1: An image of a parrot used as an input pattern

identical)

On the other hand, the second option shows a rich numeric variable that is clearly defined as a similarity or a distance measure. The distance measure here is representative and illustrates a real information of how the two patterns are similar.

Figure 1.1 shows an image of a colored parrot that could be used as either a training or a test pattern.

1.1.3 Scientific Paper Categorization

In a problem where we want to classify a scientific paper to be related to one of predefined research fields like the computation intelligence or the communication fields. In the classical methods, one could define a feature vector of word frequencies to represent the paper as a preprocessing step.

However, to choose a number of effective features that represents the scientific paper, we use the training set to determine the most important words. This could lead to a large number of features. In addition to this, when a new batch of research papers become available as a part of the old training set, we will need to redefine our set of words that are used to convert a pattern to a set of word frequencies.

If we were to convert two patterns to a single feature vector representing the

similarity between the two papers, We could have the following set of features.

1) The number of common unigrams after stemming

2) The number of common bigrams after stemming

We note hear that such simple features are only two features and they do not need to be changed in case a new batch of scientific papers become available at a later time.

1.1.4 Mushroom Problem

This problem is from the UCI Machine Learning Repository. The data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The attributes available for each pattern were the cap-shape, the cap-surface, the cap-color, the bruises, the odor, the gill-attachment, the gill-spacing, the gill-size, the gill-color, the stalk-shape, the stalk-root, the stalk-surface-above-ring, the stalk-surface-below-ring, the stalk-color-above-ring, the stalk-color-below-ring, the veil-type, the veil-color, the ring-number, the ring-type, the spore-print-color ,the population and the habitat.

Each pattern is described by 22 characters; each of them is describing an attribute. If we are going to define a feature vector for each pattern, we will need to encode such characters. Such encoding leads to either a large number of binary features or will lead to the erroneous distance relationships that we have mentioned in previous problems.

However, defining a single similarity feature vector for each two patterns eliminates such burden of encoding. One could define 11 categorical features each of them takes either the value of 0 or 1. A feature in this case will show if the two mushroom patterns have the same corresponding character or not by carrying the value of 1 or 0.

1.1.5 Shape recognition

In our previous diagram recognition work, we have defined a few and effective features to represent a shape pattern. We were however interested in adding a distance measure as a feature representing similarity between two shapes. In this problem, we have two types of features: features that are representing single patterns and the distance measure that is defined to represent the similarity between two patterns.

1.1.6 Stock Market Prediction

In this problem, we model the stock market prediction problem as a binary classification problem where we are interested in whether the stock price will rise or fall.

It is assumed that a rational measure based on this prediction is taken. Such assumption is used to compute the annual profit. In the classical classification problem, lag samples, moving averages, and other features are used to represent each time series that we are to predict its behavior in the coming year.

We propose here that dynamic time warping features could be used in the new classification framework but not with classical one. The following subsection explains time warping in brief.

Time Warping

Dynamic time warping is an algorithm for measuring similarity between two sequences which may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and decelerations during the course of one observation. DTW has been applied to video, audio, and graphics indeed, any data which can be turned into a linear representation can be analyzed with DTW. A well known application has been automatic speech recognition, to cope with different speaking speeds.

In general, DTW is a method that allows a computer to find an optimal match between two given sequences (e.g. time series) with certain restrictions. The sequences are "warped" non-linearly in the time dimension to determine a measure of

their similarity independent of certain non-linear variations in the time dimension.

One example of the restrictions imposed on the matching of the sequences is on the monotonicity of the mapping in the time dimension. Continuity is less important in DTW than in other pattern matching algorithms; DTW is an algorithm particularly suited to matching sequences with missing information, provided there are long enough segments for matching to occur.

The extension of the problem for two-dimensional "series" like images (planar warping) is NP-complete, while the problem for one-dimensional signals like time series can be solved in polynomial time.

A typical DTW algorithm could be:

```
int DTWDistance(char s[1..n], char t[1..m])
    declare int DTW[0..n, 0..m]
    declare int i, j, cost

for i := 1 to m
    DTW[0, i] := infinity
end
for i := 1 to n
    DTW[i, 0] := infinity
end
    DTW[0, 0] := 0

for i := 1 to n
    for j := 1 to m
    cost:= d(s[i], t[j])
    DTW[i, j] := cost + minimum(DTW[i-1, j ], // insertion
    DTW[i , j-1], // deletion
    DTW[i-1, j-1]) // match
    end
end
```

return DTW[n, m]

The output of DTW algorithm is a distance measure between two series and could be used as an effective similarity feature.

1.2 Introduction

1.2.1 Classical Classification Framework

In the classical classification framework, training patterns are first converted to feature vectors which are then used to train the classifier. At the point of replacing the pattern by a feature vector representing it, a significant amount of information is lost. In addition, sometimes representing the pattern by a feature vector could be problematic. For example if we would like to represent a document by a feature vector, we could use a dictionary to create a feature vector of word. This could lead to a huge number of features [1].

1.2.2 New Classification Framework

An alternative and seemingly more efficient way is to define a feature vector that represents the similarity between a pair of documents. In such case we could just define a vector that consists of a few simple and effective high level features. This similarity vector could, for example, consists of the number of common stemmed words between the pair of documents, the number of common named entities, the number of common semantic relations, and finally a binary feature showing whether the two documents were extracted from the same source or not. This suggests that a significant achievement could be acquired, if we could change the classification framework to using feature vectors that represent the similarity between a pair of patterns rather than using feature vectors that represent single patterns.

1.2.3 SVM suitability to apply the new framework

SVM is a suitable classifier for applying this new framework. In SVM, the classical kernels take two feature vectors as input (each feature vector represents a pattern) and return a real number representing the similarity between them [2]. In order to make use of high level similarity features as stated previously, a domain expert is required to invent a user defined kernel which is an algorithm that measures the similarity between two patterns without converting them to feature vectors. The domain expert is required in order to determine the contribution of each component similarity feature to the final similarity measure. This is a time consuming task since it has to be done for each problem. Moreover, a hard quantitative approach would lead to more consistent performance, and allows the use of cutting edge optimization methods.

1.2.4 Contribution

In this thesis, we propose a new kernel function that is learned statistically from data. The input of this new kernel function will be only one feature vector representing the similarity between the two input patterns. We name our new proposed kernel the SVM'ed-Kernel for a reason that will be clear.

We propose a method to automatically generate a recreated training set from the original training set. The recreated training set is then used to learn the SVM'ed-Kernel. Interestingly, the SVM'ed-Kernel will be learned as a separate SVM classification problem. Once trained, the SVM'ed-Kernel will then be used as a kernel function in the original classification SVM problem. Using the SVM'ed-Kernel, we need not define features to represent a single pattern. We will only need to define features that represent the similarity between a pair of patterns. This allows novel features to be defined that could not have been defined using the classical feature definition framework.

Moreover, a simple similarity feature between a pair of patterns could eliminate a large number of features representing a single pattern as it was shown in the example of representing a document by a feature vector. This contributes to dimensionality

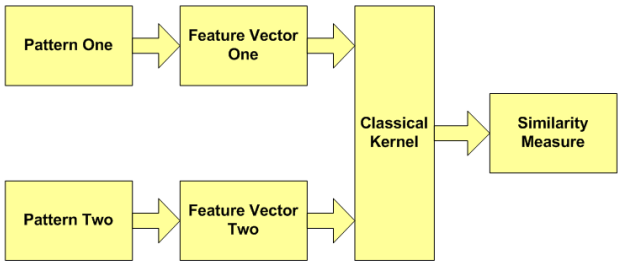

Figure 1.2: Classical Kernel block diagram

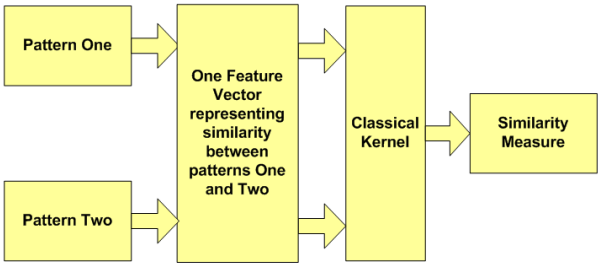

Figure 1.3: SVM'ed-Kernel block diagram

reduction. Figures 1.2 and 1.3 show the classical kernel and the SVM'ed-Kernel block diagrams respectively. In the SVM'ed-Kernel, the contribution of each similarity feature to the final similarity measure is learned statistically from the recreated training set. This eliminates the need for a domain expert, allows the definition of novel high level similarity features, and leads to optimizing the contributions of the different similarities.

The proposed kernel could be used in Natural Language Processing, Machine Vision, and Bioinformatics applications that suffer from the loss of a significant amount of information at the point where the pattern is replaced by a feature vector representing it. We tested our SVM'ed-Kernel in several classification problems and showed that it gives very promising results as compared to the traditional SVMs.

1.3 Thesis Organization

The thesis is organized as follows: In chapter 2, an introduction to Support Vector Machines is given. In chapter 3, we summarize the related work that proposed different kernel functions. The novel proposed kernel function is explained in chapter 4. We apply the SVM'ed Kernel to a toy problem in chapter 5. In chapter 6, the SVM'ed kernel is used to classify hand-drawn shapes. Th Mushroom classification problem is solved using the SVM'ed kernel in chapter 7. In chapter 8, we show how the SVM'ed kernel could be used for. We conclude and state our future work in chapter 9.

Chapter 2

Support Vector Machines

2.1 Introduction

One of the fundamental problems of learning theory is the following: suppose we are given two classes of objects. We are then faced with a new object, and we have to assign it into one of the two classes. This problem can be formalized as follows: we are given empirical data $(x_1, y_1), ...m(x_m, y_m) \in \chi \times \pm 1$ Here, χ is some nonempty set from which the patterns x_i are taken , usually referred to as the domain; the y_i are called labels.

Note that there are only two classes of patterns . For the sake of mathematical convenience they are labeled by +1 and -1 respectively. This problem is known as the binary classification problem. For example, we could be interested in classifying sheep into one of two categories. In this case, the patterns would simply be sheep.

In learning we want to be able to generalize for unseen patterns that were not available during the training phase.

2.2 Classification

In a two class classification problem, the output of the system takes only two symbolic values $y = 0, 1$ corresponding to two classes. Hence, the output of the learning machine needs to only take on two values as well.

We have a set of functions that we choose one of them during the learning process. In this case, this set of functions becomes a set of indicator functions.

In learning problems, we typically try to minimize a risk function during the learning stage. In such classification problems, the loss function takes the following form:

$$L(y, f(x, w)) = \begin{cases} 0 & \text{if } y = f(x, w), \\ 1 & \text{if } y \neq f(x, w), \end{cases} \tag{2.1}$$

Using the loss function , the risk function:

$$R(w) = \int L(y, f(x, w))p(x, y)dxdy \tag{2.2}$$

quantifies the probability of misclassification. Learning then becomes the problem of estimating the indicator function $f(x, w_0)$ that minimizes the probability of misclassification using only the training data.

2.3 Mapping to a higher dimensional space

In the 1990s, a new type of learning algorithm was developed, based on results from statistical learning theory: the Support Vector Machine (SVM). The basic idea of SVM classifiers is to map a given data set from input space into higher dimensional feature space , called dot product space, via a map function ϕ , where

$$\phi : R^N \longrightarrow F \tag{2.3}$$

This data points become linearly separable in the higher dimensional space after it was non-linearly in the original input space.

Then, it performs a linear classification in the higher dimensional space. This requires the evaluation of dot products:

$$k(x, y) = (\phi(x), \phi(y)) \tag{2.4}$$

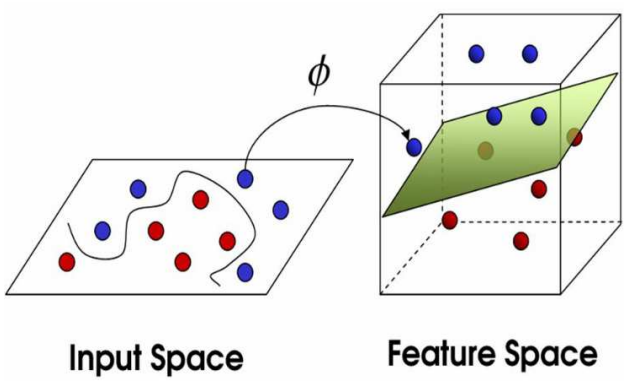

Figure 2.1: Mapping to a higher dimensional space

Where k is called the kernel function. If F is high dimensional, the right hand side of equation (2.4) will be very expensive to compute [2]. Therefore, kernel functions are used to compute the dot product in the feature space using the input parameters which means that the mapping is done implicitly. A kernel function returns a real number representing the similarity of its two input patterns. There are many types of kernels such as the RBF kernel, given by:

$$k(x, x_i) = e^{-\|x-x_i\|^2/2\sigma^2} \tag{2.5}$$

Other similar kernels are also widely used for example the Sigmoid kernel:

$$k(x, x_i) = \tanh(\kappa\langle x, x_i\rangle + \vartheta) \tag{2.6}$$

where $\kappa > 0$ and $\vartheta < 0$

and the polynomial kernel:

$$k(x, x_i) = \langle x, x_i\rangle^d \tag{2.7}$$

Figure 2.1 shows a set of patterns that are mapped to a higher dimensional feature space. The function used for the assignment of new objects to one of the two classes

is called the decision function which takes the form:

$$f(x) = \begin{cases} +1 & \text{if } \sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b > 0, \\ -1 & \text{if otherwise.} \end{cases} \tag{2.8}$$

Where, l denotes the number of training patterns
x denotes the unseen pattern vector
x_i denotes training pattern vector
y_i denotes the label of the training pattern
b denotes constant offset (or threshold)
1 and -1 are the labels of the decision classes
The parameters $\alpha_i{}'s$ are computed as the solution of a quadratic programming problem.

2.4 Discussion

Support vector machines is the learning approach originally developed by Vapnik and co-workers. The aim of support vector machines is to devise a computationally efficient way of learning good separating hyper planes in a high dimensional feature space.

The simplest model of SVM is called the maximal margin classifier, it works only with data which are linearly separable in the feature space and hence can not be used in real world situations. The strategy is implemented by reducing it to a convex optimization problem, minimizing a quadratic function under linear inequality constraints.

The maximal margin classifier optimizes this bound by separating the data by the maximal margin hyper plane. Since it does not depend on the feature space, it can be used for any kernel induced feature space. Note that in linear classifiers there is an inherent degree of freedom in which we can rescale the hyper plane. The functional margin will change but the geometric one will not. We have $(w, x) + b = 1$ and $(w, x) + b = -1$ as the 2 equations of the 2 planes at the sides of the margin. It

was shown that the margin will be equal an expression which contains the $\|w\|$ in the denominator so if we manage to minimize $\|w\|$ then we have managed to maximize the margin. Using Lagrange multipliers, we were able to find out that the hypothesis was represented as linear combination of the training points, also the application of the optimization theory has leaded us to the dual representation. Only inputs x_i for which the functional margin is one, and these therefore lie closest to the hyper plane, have non-zero Lagrange multipliers . All other Lagrange multipliers vanish. Hence in the expression of the weight vector only these points are involved [2].

The fact that only a subset of the Lagrange multipliers is non zero is referred to as sparseness and means that the support vectors contain all the information necessary to reconstruct the hyper plane even if all other points were removed. The smaller the number of support vectors, the better generalization can be expected. The maximal margin classifier does not attempt to control the number of support vectors and in practice there are very few support vectors. The only degree of freedom in the maximal margin algorithm is the choice of kernel, any prior knowledge we have about the problem can help us choose a parameterized kernel family and then model selection is reduced to adjusting the parameters. For most classes of kernels, for example polynomial and Gaussian, it is always possible to find a parameter for which the data become separable, however forcing separation of the data can easily lead to over fitting particularly when noise is present in the data. In this case, the outlier will be characterized by a large Lagrange multiplier.

Weaknesses of the maximal margin classifier are that it is sensitive to noise, considers only two classes and is not designed to achieve sparse solutions. The maximal margin classifier is an important concept as a starting point for the analysis and construction of more sophisticated support vector machines but it can not be used in many real world applications. If the data is noisy there will be no linear separation in the feature separation unless we are using very powerful kernels and by that we will overfit the data [2].

2.5 Kernel Choice

Support vector machines have been applied to many real world applications. Collaboration between the system designer and the domain practitioner can be used to refine the data and to choose the kernel. Of course there are some standard choices like Gaussian or polynomial but sometimes they are ineffective. By implicitly defining a feature space, the kernel provides the description language used by the machine for viewing the data. Typically, the choice of kernels will be a family of kernels parameterized by a hyper parameter as for example in Gaussian kernels the parameter sigma needs to be determined [2]. The choice of kernel parameters is usually in response to the data. In the absence of reliable criteria, applications rely on the use of cross validation to set such parameters.

For a kernel function to be an inner product in the feature space, it must satisfy the Mercer theorem. A kernel function K must be continuous, symmetric, and have a positive definite gram matrix. Such a K means that there exists a mapping to a reproducing kernel Hilbert space (a Hilbert space is a vector space closed under dot products) such that the dot product there gives the same value as the function K. If a kernel does not satisfy Mercer's condition, then the corresponding QP may have no solution [2].

Chapter 3

Related Work

Many Kernels have been proposed in the SVM literature. We divide the related work into general kernels and specific user-defined kernels. The general kernels are not defined for a specific problem. On the other hand, the user defined kernels are domain dependent and they are defined specifically for the problem at hand. Our proposed SVM'ed-Kernel falls in the general kernels class.

3.1 General Kernels

3.1.1 Evolutionary Kernels

From among the general proposed kernels, Thadani et al [3] creates a kernel function suitable for the training data using a genetic algorithm mechanism. They showed that their genetic kernel has good generalization abilities when compared with the polynomial and the radial basis kernel functions.

3.1.2 Autocorrelaton Kernel

Kong et al [4] proposed the autocorrelation kernel by borrowing this concept from signal processing. The autocorrelation functions give comparable results to the RBF kernel when used to classify some UCI datasets.

3.1.3 Chebyshev Kernel

Ye et al [5] proposed an orthogonal Chebyshev kernel function. Chebyshev polynomials are first constructed through Chebyshev formulae. Then based on these polynomials Chebyshev kernels are created satisfying Mercer condition. They showed that it is possible to reduce the number of support vectors using this kernel. In addition, they require the features to be normalized from -1 to 1.

3.1.4 Wavelet Kernel

George et al [7] proposed a Sinc-Cauchy hybrid wavelet kernel and shows that it is admissible which means that it is positive definite [2]. They used it for the classification of Cardiac Single Photon Emission Computed Tomography images and Cardiac Arrhythmia signals. Their experimental results showed that promising generalization can be achieved with the hybrid kernel compared to conventional kernels.

3.1.5 Mahalanobis Distance Kernel

Wang et al [9] proposed the Weighted Mahalanobis Distance Kernels. They first find the data structure for each class in the input space via agglomerative hierarchical clustering and then construct the weighted Mahalanobis distance kernels which are affected by the size of clusters they reside in. They showed that, although WDM kernels are not guaranteed to be positive definite or conditionally positive definite, satisfactorily classification results can still be achieved because regularizes in SVMs with WDM kernels are empirically positive in pseudo-Euclidean spaces.

3.1.6 Log Kernel

Boughorbel et al [6] proposed the log kernel which seemed particularly interesting for images. They proved that the log kernel is conditionally positive definite. Moreover, they showed from experimentations that using conditionally positive definite kernels allows us to outperform classical positive definite kernels.

3.2 User-Defined Kernels

3.2.1 String Kernels

From among the specific user-defined kernels, XU et al [10] proposed using the weighted Levenshtein distance as a kernel function for strings. They used the UCI splice site recognition dataset for testing their proposed specific kernel which got the best results in this problem.

3.2.2 RNA Classification

Wu et al [12] proposed a new user-defined kernel for RNA classification. They showed that the new kernel takes advantage of both global and local structural information in RNAs. Their experimental results showed that the new kernel outperforms existing kernels when used to classify non-coding RNA sequences.

3.2.3 Document Kernels

Siolas et al [11] proposed using a new metric between documents based on a priori semantic knowledge about words. They incorporated this metric into the definition of radial basis function which improved the performance.

3.2.4 Gene Sequences

Yan et al [13] proposed the position weight subsequences kernel (PWSK) that could be used for identifying gene sequences. This kernel was used for splice site identification and the performance was better than that of the string subsequences kernel (SSK). Cuturi et al [14] proposed a mutual information kernel for strings which borrows techniques from information theory and data compression. They showed that their kernel reported encouraging classification results on a standard protein homology detection experiment.

3.3 Other Kernels

3.3.1 Learning Kernels by boosting

Crammer et al [26] proposed learning kernel operators from other less accurate base kernels. They used the boosting paradigm to construct the learned kernel. Their approach is suitable in trusductive learning tasks where the test data distribution is different from the training data distribution. They showed that On the USPS dataset, the performance of the Perceptron algorithm with learned kernels is systematically better than a fixed RBF kernel.

3.3.2 Comparing text Documents

Lodhi et al [27] introduced a novel kernel for comparing two text documents. The kernel is an inner product in the feature space consisting of all subsequences of length k. A subsequence is any order sequence of k characters occurring in the text though not necessarily contiguously. These subsequences were weighted by some decay factor of their full length in the text, hence putting some emphasis on contiguous characters. The paper showed how such dot product is computed without explicit conversion to feature vectors which is inefficient.

3.3.3 Comparing Sequential Data

Rieck et al [28] proposed an algorithm for computation of similarity measures for sequential data. The algorithm uses suffix trees for efficient calculation of various kernel functions. Its worst-case run-time is linear in the length of sequences and independent of the underlying embedding language, which can cover words, k-grams or all contained subsequences. Experiments with network intrusion detection, DNA analysis and text processing applications demonstrate the utility of distances and similarity coeffi- cients for sequences as alternatives to classical kernel functions.

3.3.4 Mimicking Multilayer Neural Networks

Cho et al [29] introduced a new family of positive-definite kernel functions that mimic the computation in large, multilayer neural nets. These kernel functions can be used in shallow architectures, such as support vector machines (SVMs), or in deep kernel-based architectures that they call multilayer kernel machines (MKMs). They evaluated SVMs and MKMs with these kernel functions on problems designed to illustrate the advantages of deep architectures. On several problems, they obtained better results than previous, leading benchmarks from both SVMs with Gaussian kernels as well as deep belief nets.

3.4 The Proposed Kernel

Our proposed kernel falls in the general kernels class while having the ability of defining similarity features which have been only used in specific user defined kernels. Moreover, it does not need a domain expert to determine the contribution of each similarity feature to the similarity measure since the kernel is learned statistically from data which is extracted automatically from the original training set.

Chapter 4

The Support Vector Machined Kernel

4.1 Learning Dependencies

Modern Science and Engineering are based on using first-principle models to describe physical, biological, and social systems. Such approach starts with a basic scientific model (e.g. Newton's law of mechanics) and then builds upon them various applications. However, in many applications the underlying first principles are unknown or the systems under study are too complex to be mathematically described. Fortunately, with the growing use of low-cost computers and sensors for data collection, there is a great amount of data being generated by such systems. Such data could be used derive models by estimating useful relationships between system variables (unknown input-output dependencies).

4.2 Drawing Conclusions from the data

It is essential to realize that the process of learning dependencies from data is only one step in the general experimental procedure used by scientists, engineers, doctors, social scientists and others who apply statistical methods to draw conclusions from the data. The general experimental procedure adopted in classical methods involves

the following steps, adapted from Dowdy and Weardan (1991):

1. State the problem
2. Formulate the hypothesis
3. Generate the data
4. Perform preprocessing
5. Estimate the model
6. Interpret the model

We briefly explain each of them in the following subsections.

4.2.1 State the problem

Most data modeling studies are performed in a particular application domain. Hence, domain-specific knowledge and experience are usually necessary in order to come up with a meaningful problem statement.

4.2.2 Formulate the hypothesis

The hypothesis in this step specifies an unknown dependency, which is to be estimated from experimental data. At this step, the modeler usually specifies a set of input output variables for the unknown dependency and, if possible, a general form of it. This step requires cooperation between domain expert and statistical modeler.

4.2.3 Generate the data

This step is concerned with how the data is generated. When such generation is under control of the modeler, the setting is known as the designed experiment. However, when the modeler cannot influence the data generation process, the setting is called observational setting. It is very important to make sure that the data used for training and that used in subsequent real operation come from the same (unknown) sampling distribution. If this is not the case, the predictive models estimated from the training data cannot be used for prediction with the future data.

4.2.4 Perform preprocessing

Data preprocessing includes the removal of outliers. Outliers could be a result of measurement errors or a natural abnormal existence. These outliers can seriously affect the model produced later in step 5. However, some approaches depend on robust modeling methods that are insensitive to such rogue patters. Furthermore, data preprocessing involves variable scaling and different types of encoding techniques. Such application dependent encoding techniques usually achieve dimensionality reduction by providing a small number of informative features for subsequent data modeling.

There is usually a strong connection between step 4 and the subsequent model estimation step. For example, consider the task of variable scaling. The problem of scaling is that different input variables have different units and therefore different scales. For some modeling methods (e.g. classification trees) this does not cause a problem. However, other methods (e.g. distance based methods) are very sensitive to the chosen scale of input variables. Hence, each input variable needs to be rescaled by the standard deviation of its values. However, independent scaling of variables may lead to suboptimal representation for many learning methods.

Encoding often includes selection of a small number of informative features from a high dimensional data. This is known as feature selection in pattern recognition. It may be argued that good preprocessing/data encoding is the most important part in the whole procedure because it provides a small number of informative features, thus making the task of estimating dependency much simpler . Indeed, the success of many application studies is usually due to clever preprocessing/data encoding scheme rather than the learning method used.

4.2.5 Estimate the model

Each hypothesis in step 2 represents an unknown dependency between inputs and outputs. In this step, such dependency is quantified using the available data and a priori knowledge about the problem. Examples of methods used to estimate the model includes neural networks and support vector machines.

4.2.6 Interpret the model

The estimated models are sometimes used for (human) decision making. Therefore, such models need to be interpretable in order to be useful because humans are not likely to base their decision on complex black box models. Note that the goals of accurate prediction and interpretation are different because interpretable models would be necessarily simple but accurate predictive models may be quite complex.

4.3 Characterization of Variables

Each of the input and output variables can be of several different types. The two most common types are categorical and numeric variables. numeric variables includes real-values or integer variables.

A numeric variable has two important features: Its values have an order relation and a distance relation defined between any two values. Whereas, categorical variables have neither order nor distance relations defined. Thus, any two values of a categorical variable can only be either equal or unequal.

There are two other common types of variables: periodic and ordinal. A periodic variable is a numeric variable where the distance relation is undefined. Examples are days of the week, month, or year.

In contrast, ordinal variables are categorical variables for which the order relation is defined but no distance relation. Examples are gold, silver, and bronze medal position in a sport competition. The reason for not having a distance definition for ordinal variables is that these values are subjectively defined by humans in a particular context. Furthermore, there is no clear boundary between the two closest values. Instead ordinal values denote overlapping sets.

4.4 Beyond Preprocessing

We have stated that there is usually a very strong connection between step 4 and step 5 in the process of learning dependencies from data. Additionally, we have showed how

variable scaling and rogue pattern removal could affect model estimation. Moreover, we have showed that selecting a small number of effective features is vital to the success of the whole process of learning. We note here that if we are to change the representation of such input features, we will necessarily need to develop a novel model estimator that supports the new representation.

4.4.1 A new learning framework

We define a new framework that does not require the conversion of a pattern to a single feature vector. Instead, it will require that each pair of patterns could be converted to a feature vector of different similarity measures.

However, in some problems (for example the shape recognition problem), we already have a set of effective features. Therefore, our new model should support the old features. In other words, it will be optimum if the new proposed framework could assimilate both type of features in a single unified framework.

In the following section, we will propose a novel kernel function that will allow the use of the novel feature definitions while being able to also use the old defined features.

4.5 Approach

The SVM'ed-Kernel could be used in any machine learning task that requires a kernel function. In this thesis, we illustrate its use as a kernel function for support vector machine classification problems.

Internally, the SVM'ed-Kernel is constructed as a support vector machine classification problem. Therefore we have two SVMs; the first one is the original SVM classification problem which we will call it the original SVM, while the other is the one used as a kernel function which we call it the SVM'ed-Kernel.

This SVM'ed-Kernel will be trained using a recreated training set extracted from the original one. The steps to create and use the SVM'ed-Kernel are:
A. Define a feature vector representing the similarity between a pair of patterns.

B. Automatically generate the recreated training set from the original one.
C. Train the SVM'ed-Kernel as a normal classification problem using the recreated training set in *B*.
D. Use the trained SVM'ed-kernel as a kernel function in the original SVM problem.
E. Train the original SVM using the SVM'ed-Kernel.

We now explain each step in details.

4.5.1 Similarity Feature Vector Definition

In step *A*, we define a feature vector that represents the similarity between two patterns. For example in a text categorization classification problem where we need to classify a document according to whether it is related to either sport or politics. One could define a similarity feature vector of two features. The first feature could be the number of common words after stemming, while the second one could be the number of common semantic relations.

4.5.2 Recreated Set Generation

In step *B*, assume that we have an original simple training set similar to that in Table 4.1. To create the recreated training set that will be used to train the SVM'ed-Kernel, we select every pair of patterns from the original training set (order is not important). So we have pattern 1 and pattern 2, pattern 1 and pattern3, pattern 2 and pattern 3, pattern 2 and pattern 4, and so on. We label each pair as being matching (1) if the two patterns have the same label in the original training set or not matching (-1) if they have different labels. Table 4.2 illustrates the recreated training set. One can see here that the recreated training set is of larger size than the original training set.

4.5.3 SVM'ed Kernel Training

In step *C*, we first convert each pair in the recreated training set, created in step *B*, to a feature vector using the similarity feature vector definition we have defined in step

Table 4.1: The original training set

patterns	class label 1 or -1
pattern 1	1
pattern 2	-1
pattern 3	1
pattern 4	-1

Table 4.2: The recreated training set

patterns	class label 1 or -1
pattern 1 and pattern 2	-1
pattern 1 and pattern 3	1
pattern 1 and pattern 4	-1
pattern 2 and pattern 3	-1
pattern 2 and pattern 4	1
pattern 3 and pattern 4	-1

A. After that, we train the SVM'ed-Kernel as a normal SVM classification problem using the recreated training set and any arbitrary kernel. The output of this SVM classifier will be a label indicating whether the two input patterns are matching or not. After training, we save the SVM trained as our SVM'ed-Kernel after removing the decision component from the function in equation (2.8), to be in the form

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b \tag{4.1}$$

The decision component was removed because we are interested in the real value returned from (4.1), which represents how confident we are in the match. A larger returned value represents a better match (high similarity) and vice versa. Figure 4.1 shows three pairs, from the recreated training set, and their locations from the decision boundary of the SVM'ed-Kernel. When substituting in equation (4.1), pair one's similarity feature vector will return a positive number indicating high similarity

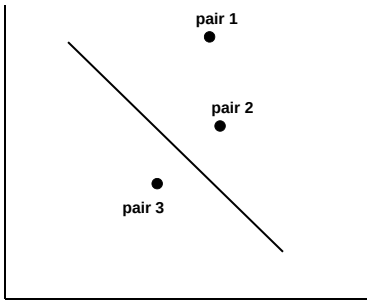

Figure 4.1: Three pairs and their locations from the decision boundary of the SVM'ed-Kernel

between this pair. On the other hand, pair two's similarity feature vector will return a smaller positive number indicating that this pair is less similar than pair one. Finally, pair three's vector will return a negative number indicating low similarity between this pair.

The training patterns in the recreated training set were used to determine the maximal margin classifier. The distance of a new pair from the maximal margin classifier decision boundary is a direct measure of the similarity between the two patterns of this pair due to the way we formed the recreated training set in B and the definition of the similarity feature vector in A.

4.5.4 Using the SVM'ed Kernel

In step D, we use our trained SVM in C as our kernel function in the original SVM problem. In our work we have added an offset to the output of the SVM'ed-Kernel so that the similarity measure returned becomes always positive. Such offset could be determined using cross validation.

4.5.5 Training the Original SVM

In step E, we train our original SVM using our SVM'ed-Kernel and the original training set. After training the original SVM, the original SVM model is ready for the real operation phase. We conclude here that the original SVM classifier component

does not require the pattern to be extracted to a feature vector on its own. It takes the form:

$$f(pattern) = \begin{cases} +1 & \text{if } \sum_{i=1}^{l} \alpha_i y_i SVK(pattern, pattern_i) + b > 0, \\ -1 & \text{if otherwise.} \end{cases} \tag{4.2}$$

where SVK is the SVM'ed-Kernel in the form:

$$SVK(pattern_x, pattern_y) = \sum_{j=1}^{l'} \alpha_j y_j k(m(pattern_x, pattern_y), m_j) + b' \tag{4.3}$$

Where, m_j denotes a similarity feature vector of a support vector pattern for the SVM'ed-Kernel.
$m(x, y)$ denotes the feature vector representing similarity between $pattern_x$ and $pattern_y$.
y_i denotes the label of the i^{th} pattern in the original training set.
y_j denotes the label of the j^{th} pair in the recreated training set.
b and b' denote constant offsets (or thresholds).
k denotes an arbitrary kernel function.
l and l' denote the number of training patterns in the original and the recreated training sets respectively.
The parameters $\alpha_i's$ and $\alpha_j's$ are computed as the solutions of quadratic programming problems.

Chapter 5

Toy Problem

5.1 Problem Definition

5.1.1 Introduction

In this toy problem, we show that the SVM'ed-Kernel is capable of using the already existing feature definitions. This problem is a hand drawn circle-triangle classification problem. The input is a hand drawn shape and the output is its classification as a circle or a triangle. We assumed that the unseen shape should be either a circle or a triangle.

5.1.2 Data Set

The data set used was formed from the circles and triangles patterns created by Refaat et al [15] after increasing its size to be 222 circles and 222 triangles. We divided the dataset set randomly into 400 patterns for training and 44 patterns for testing. We compared our SVM'ed-Kernel to different kernels with different parameters. We used SVM Light [16] in our simulations.

5.1.3 Previous Work

In our previous work [15], a number of basic features were extracted from each shape for the purpose of serving as inputs to the classifier. The features were selected to be size and orientation independent.

5.1.4 Basic Shape Features

The basic shape features used are Ach, the area of the convex hull;Pch, its perimeter;Aer, the area of the rectangle enclosing the convex hull of the shape and having the minimum area;Alq, the area of the maximum area inscribed quadrilateral that fits inside the convex hull of the shape [17];Alt, the area of the maximum area inscribed triangle that fits inside the convex hull of the shape [17];Plt, its perimeter;Her, the height of the rectangle enclosing the convex hull of the shape and having the minimum area; and Wer, its width.

5.1.5 Feature Extraction

The features used in the SVM model were: Pch^2/Ach (Thinness ratio), this feature distinguishes in particular circles from other shapes. The thinness ratio of a circle is minimal, since it is the planar figure with the smallest perimeter enclosing a given area [24]; Ach/Aer this feature is useful for distinguishing rectangle shapes from other shapes. The Ach/Aer ratio will have values near unity for rectangles [24]; Alq/Aer, this feature is good for distinguishing rectangles from other shapes. Alq/Aer ratio will have values near unity for this shape [24]; Alq/Ach, this feature helps distinguishing ellipses from other shapes. Alq/Ach ratio will have values near 0.7 for ellipses and bigger values for other shapes [24]; Alt/Alq, this feature distinguishes diamonds from other shapes. The Alt/Alq ratio will have values between 0.5 and 0.6 for diamonds and bigger ones for other shapes [24]; Alt/Ach, This feature helps distinguishing triangles from other shapes. Alt/Ach will have values near unity for triangles and smaller values for other shapes [24]; Plt/Pch, this feature distinguishes triangles from other shapes [24]. Plt/Pch will have values near unity for triangles and smaller values for other shapes; Her/Wer, this feature can distinguish lines from other shapes [24].

5.1.6 Similarity Feature Vector

The similarity feature vector definition that is defined for the sake of the SVM'ed-Kernel was defined simply to be the subtraction of the two feature vectors of the two input patterns and taking the norm of each subtraction result.

Subtraction between two values is a measure of similarity. For example if $f_1 = 5$ in the first shape and $f_1 = 5$ in another one, the subtraction of these features will give a zero. A zero will mean that the two patterns are equally similar with respect to such feature.

On the other hand, if $f_1 = 10$ and $f_1 = 2$ for two shapes. The magnitude of subtraction will result in 8 giving an indication that the two patterns are different with respect to f_1.

It is interesting to mention here that we could define either similarity measures or dissimilarity measures. Subsequently, when we learn the SVM'ed kernel, the weights of learned SVM will reflect whether these features are similarity or dissimilarity measures.

In this problem, we did not define new similarity features that make use of the benefit of the SVM'ed- Kernel to evaluate our kernel power of making use of the already existing feature definition that was defined for a single pattern.

5.2 Results

For the hand drawn circle-triangle problem, we used the SVM'ed-Kernel with just subtracting the two feature vectors of the two input patterns as shown in figure 5.1. We also used the polynomial kernel (Poly) with the d parameter set to 2, 3 and 4; the RBF kernel with the gamma parameter set to 1, 2, 0.1 and 0.01; and the linear kernel. Table 5.1 shows the testing results of the circle-triangle problem. Acc. and SVs stands for accuracy on the test set and number of support vectors respectively.

The SVM'ed-Kernel gives comparable testing accuracy to all other kernels with respect to the test set size and gives the least number of support vectors.

The SVM'ed-Kernel has only 6 support vectors whereas all other kernels have at

Table 5.1: Hand-drawn circle-triangle problem testing results

Kernel Used	Acc.	XiAlpha *recall*	XiAlpha *precision*	SVs
SVM'ed	97.73%	99.5%	99.5%	6
Poly d=2	97.73%	91.5%	91.5%	34
Poly d=3	97.73%	90.0%	90.0%	40
Poly d=4	97.73%	81%	81%	76
RBF g=1	100%	97.5%	100%	68
RBF g=2	100%	97.0%	100%	84
RBF g=0.1	100%	98.5%	98.99%	42
RBF g=0.01	100%	94.5%	94.97%	27
Linear	97.73%	86%	86%	56

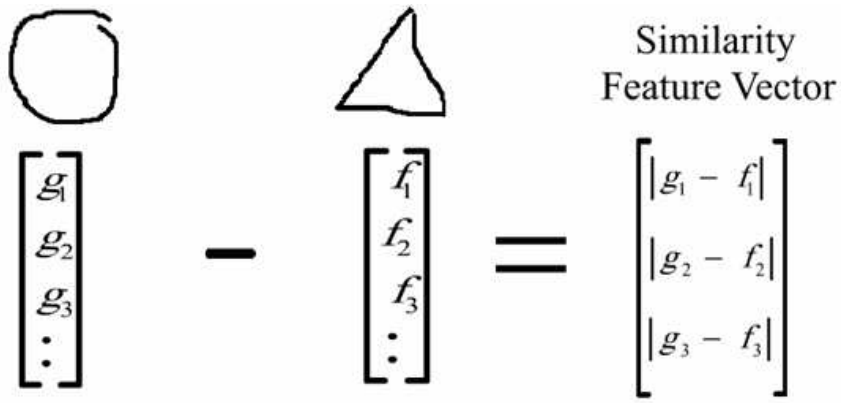

Figure 5.1: A block diagram for subtracting two feature vectors to extract a similarity vector

least 27 support vectors. This shows that the SVM'ed-Kernel ability of generalization outperforms all other classical kernels [2].

In addition, the SVM'ed-Kernel outperforms all other kernels in the Joachim Xi alpha [18] estimate of the recall and shows very promising Joachim Xi alpha of the precision.

We note here that we did not make use of the benefit of defining features of similarity between a pair of patterns but only showed that the SVM'ed-Kernel is capable of using the original feature definitions that were already defined to all problems before.

Chapter 6

Shape Recognition

6.1 Introduction

Structured diagrams (flow charts, Markov chains, module dependency diagrams, state diagrams, block diagrams, UML, graphetc) have become indispensable tools for the business world. They are prevalent in all types of documents, whether business reports, research publications, white papers, etc.

Most people who need to create such diagrams use structured graphics editors such as Microsoft Visio [23]. Structured graphics editors are extremely powerful and expressive but they can be cumbersome to use. We have performed extensive experiments comparing the times of hand-drawn structured diagrams with those entered using a tool like Visio. The result is that the hand-drawn diagrams take on average 90% less time. This suggests that an automated solution, based on hand-written diagram recognition, will save a significant amount of time for the user, and hence will be economically very beneficial.

Recently, applications have been developed that use online systems running on pen-input PCs that allow users to create diagrams by drawing on the PC tablet. These systems use the so-called online diagram-recognition approach. The other harder type is the off-line diagram recognition approach, which is based on processing the image as a whole after it has been completely drawn. The online approach makes use of the sequence of strokes, and this is a very facilitating piece of information. While there

have been several works in the literature on online recognition, the progress of offline diagram recognition is still very minimal. This is partly due to the difficulty of the problem.

In our previous work, we proposed a system of off-line diagram recognition. The proposed model consists of all possible components of a diagram recognition system, such as segmentation, feature extraction, classification, and redrawing and repositioning.

For the first component, we need to segment the diagram into standalone shapes. For that, our approach utilizes Line Primitive Extraction by Interpretation of Line Continuation [8]. We proposed a novel adjustment to this algorithm in which we enhance the output of segmentation by reattaching shapes that have been erroneously divided.

In addition, we proposed a novel algorithm that detects hand-drawn lines and distinguishes them from shapes. In the second component size and orientation independent features should be extracted from each shape. We proposed a group of effective features that characterize the different shapes.

The third component is the classifier, which is for the purpose of recognizing the shapes based on the extracted features. Here we used support vector machines (SVM) [2] to classify the shapes. Finally, in the last component, shapes are redrawn in the same relative position and with the same drawn size to produce a professional diagram similar to that produced by structured graphics editors currently on the market.

In this application, we use the support vector machined kernel function in an SVM setting to enhance the accuracy of the shape recognition stage of our previous work.

Figures 6.1,6.2,6.3,6.4 and, 6.5 show different hand-drawn shapes samples from the training set.

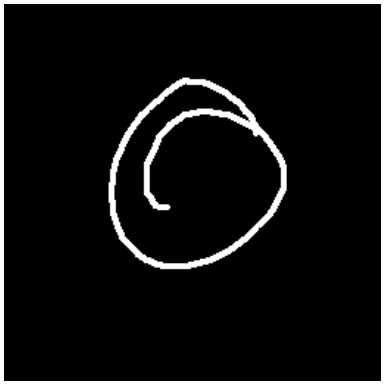

Figure 6.1: Hand-drawn circle

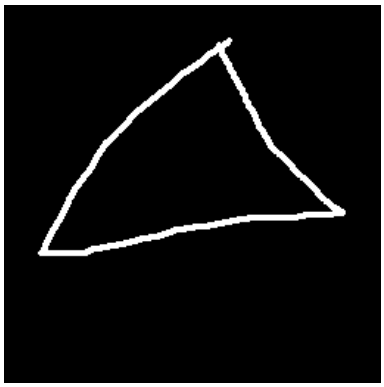

Figure 6.2: Hand-drawn triangle

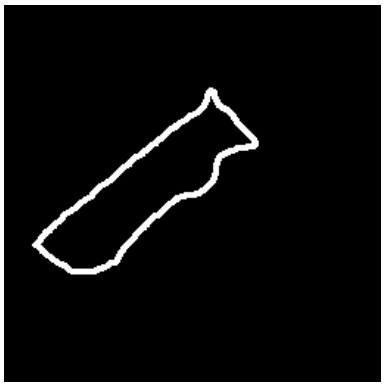

Figure 6.3: Hand-drawn rectangle

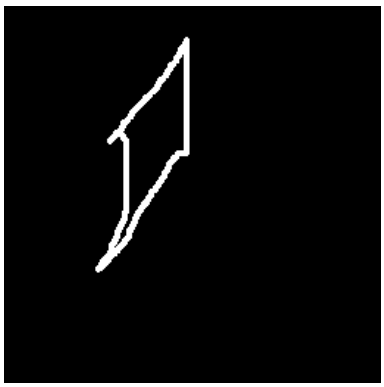

Figure 6.4: Hand-drawn diamond

Figure 6.5: Hand-drawn ellipse

6.2 Problem Definition

6.2.1 Previous Work

In our previous work [15], a number of basic features were extracted from each shape for the purpose of serving as inputs to the classifier. The features were selected to be size and orientation independent.

6.2.2 Basic Shape Features

The basic shape features used are Ach, the area of the convex hull;Pch, its perimeter;Aer, the area of the rectangle enclosing the convex hull of the shape and having the minimum area;Alq, the area of the maximum area inscribed quadrilateral that fits inside the convex hull of the shape [17];Alt, the area of the maximum area inscribed triangle that fits inside the convex hull of the shape [17];Plt, its perimeter;Her, the height of the rectangle enclosing the convex hull of the shape and having the minimum area; and Wer, its width.

6.2.3 Feature Extraction

The features used in the SVM model were: Pch^2/Ach (Thinness ratio), this feature distinguishes in particular circles from other shapes. The thinness ratio of a circle is minimal, since it is the planar figure with the smallest perimeter enclosing a given area [24]; Ach/Aer this feature is useful for distinguishing rectangle shapes from other shapes. The Ach/Aer ratio will have values near unity for rectangles [24]; Alq/Aer, this feature is good for distinguishing rectangles from other shapes. Alq/Aer ratio will have values near unity for this shape [24]; Alq/Ach, this feature helps distinguishing ellipses from other shapes. Alq/Ach ratio will have values near 0.7 for ellipses and bigger values for other shapes [24]; Alt/Alq, this feature distinguishes diamonds from other shapes. The Alt/Alq ratio will have values between 0.5 and 0.6 for diamonds and bigger ones for other shapes [24]; Alt/Ach, This feature helps distinguishing triangles from other shapes. Alt/Ach will have values near unity for triangles and smaller values for other shapes [24]; Plt/Pch, this feature distinguishes triangles from

other shapes [24]. Plt/Pch will have values near unity for triangles and smaller values for other shapes; Her/Wer, this feature can distinguish lines from other shapes [24].

6.2.4 Similarity Feature Vector

For the SVM'ed Kernel, we decided to make use of the features already designed before together while adding a novel high level similarity feature to show the power of the novel proposed kernel.

The similarity feature vector definition that is defined for the sake of the SVM'ed-Kernel was defined simply to be the subtraction of the two original feature vectors of the two input patterns. However, we have added a single similarity feature that represents the similarity between a pair of patterns. This feature was a modified chain code distance measure [22].

6.3 Results

6.3.1 Experimental Setup

In our experiments, we trained the system using hand-drawn circles, triangles, rectangles, diamonds and ellipses from Refaat *et al* data set (2008) [15].

We have added some new shapes to the set to increase its size. We divided the new extended data set into 750 patterns for training and we kept two test sets unseen.

The first test set is a normal one of 236 patterns while the other one consists of 234 hard patterns. In the hard test set the shapes may be drawn similar to more than one shape class and it is the responsibility of the model to discover its true class. The hard test set was created in order to measure our models' robustness to hard patterns or shapes drawn carelessly.

Figures 6.6,6.7 6.8,6.9,6.10, and 6.11 show samples of hard patterns.

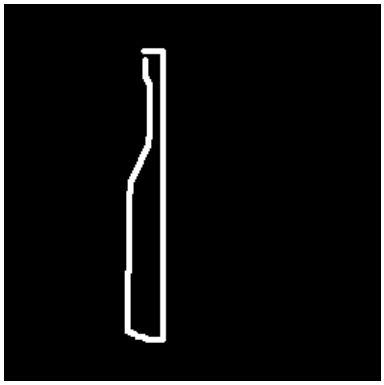

Figure 6.6: Sample of hard pattern no 1

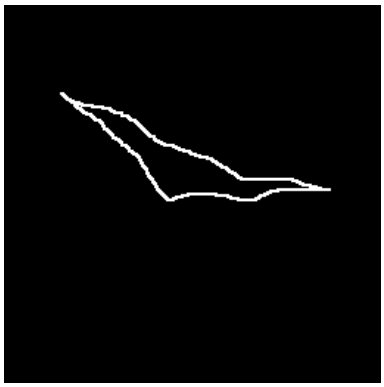

Figure 6.7: Sample of hard pattern no 2

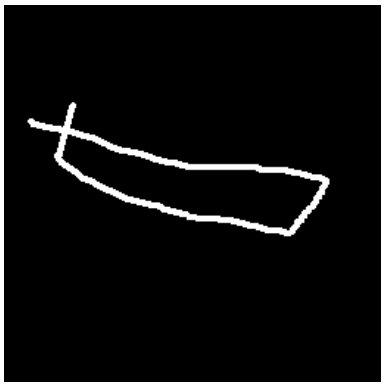

Figure 6.8: Sample of hard pattern no 3

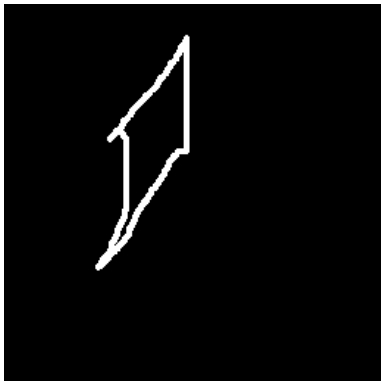

Figure 6.9: Sample of hard pattern no 4

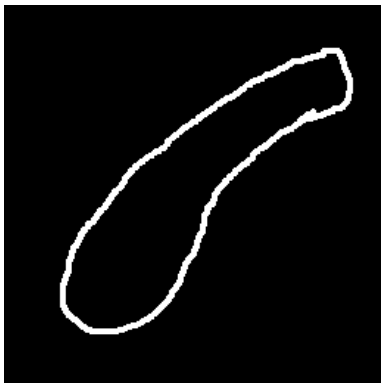

Figure 6.10: Sample of hard pattern no 5

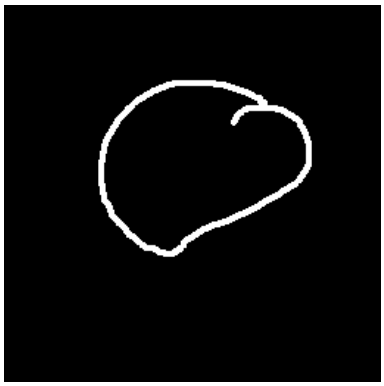

Figure 6.11: Sample of hard pattern no 6

Table 6.1: Testing accuracies for the hand-drawn shape recognition problem

Test Set-Model	RBF gamma = 2 *pairwise*	Refaat 2008	SVM'ed Kernel *pairwise*
normal	81.355%	88.135%	92.796%
hard	61.96%	69.23%	85.89%

6.3.2 Muli-Classification

We used the SVM'ed Kernel with the pairwise classification method [2] to handle the multiclass problem. A recreated set was generated from the training set of each pair of classes. Each recreated set was then used to train an SVM'ed Kernel which was used subsequently as a kernel function for the corresponding binary classifier.

All SVM'ed Kernels use the rbf kernel with the gamma parameter set to 2. We did not perform any tuning for the gamma of the rbf kernel used by the SVM'ed Kernels.

6.3.3 Simulation Results

We compared the test accuracy of the SVM'ed Kernel to that achieved by Refaat *et al* (2008) SVM model and also to that achieved by using the rbf kernel with the pairwise classification method. In the last case, by trial and error, the gamma parameter was chosen to be set to 2. We used SVMlight [16] in all our simulations. Table 6.1 shows the testing accuracies of the three models for both the normal and the hard sets.

The testing results showed that the SVM'ed kernel outperforms Refaat *et al* 2008 [15] in both the normal and the hard sets by about 4.6% and 16.5% respectively. The reason of this significant gain was that the SVM'ed kernel used the modified chain code similarity measure. This mutual feature could not have been used by the classical kernels because it represents a pair of patterns rather than only one. In addition, the SVM'ed kernel did not neglect the predefined classical features which made it act as a statistical integrator of all information about the task of shape recognition.

Chapter 7

Mushroom Problem

7.1 Problem Definition

In the second classification problem, we used the Mushroom problem from the UCI Machine Learning Repository. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

The attributes available for each pattern were the cap-shape, the cap-surface, the cap-color, the bruises, the odor, the gill-attachment, the gill-spacing, the gill-size, the gill-color, the stalk-shape, the stalk-root, the stalk-surface-above-ring, the stalk-surface-below-ring, the stalk-color-above-ring, the stalk-color-below-ring, the veil-type, the veil-color, the ring-number, the ring-type, the spore-print-color ,the population and the habitat.

Each pattern is described by 22 characters; each of them is describing an attribute. If we are going to define a feature vector for each pattern, we will need to encode such characters. Using the SVM'ed-Kernel, we do not need to define a feature vector for each pattern. We defined the similarity feature vector as a binary vector of 22 features. Feature takes the value of one if attribute is the same in the two input patterns, otherwise it takes zero. This eliminated the burden of encoding the character attributes using an elegant definition of the similarity feature vector in a binary simple

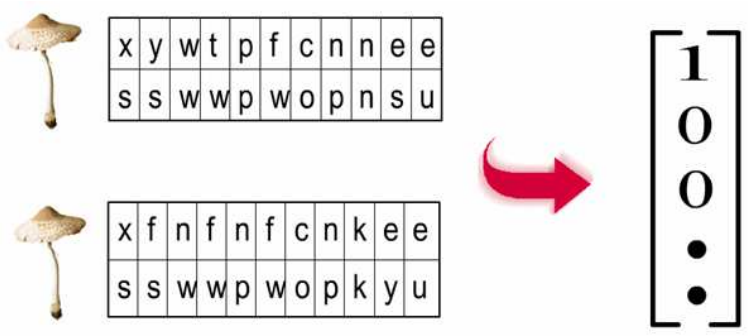

Figure 7.1: A block diagram showing the similarity feature vector definition for the mushroom problem

form.

Figure 7.1 shows a block diagram for the similarity feature definition. In this case the first feature is assigned to the value of 1 because both mushroom patterns have the same first attribute. Whereas, the second and the third feature values are assigned the value of 0 since the second and the third attributes are different in the two Mushroom patterns. One could complete the similarity feature vector by comparing the two input patterns.

While creating the similarity feature vector, we handle the missing attributes [19] in the dataset by putting the value of one if both attributes are missing. If only one attribute is missing we put zero. We divided the data set into 67% for training and 33% for testing. Interestingly, the recreated training set had more than twenty million combinations, so we have randomly sampled about 11% of this recreated training set to train our SVM'ed-Kernel due to size limitations in the SVM Light package.

In this problem the SVM'ed-Kernel uses internally the RBF kernel with the Gamma parameter set to two. We note here that any kernel could be used by the SVM'ed-Kernel.

7.2 Results

In the Mushroom problem, Kim et al [19] reported 99.51%, 96.61%, and 99.53% ten-fold cross validation accuracies using different SVM variations to avoid training with

Table 7.1: Testing Results for the Mushroom problem using the SVM'ed Kernel

Accuracy	Precision	Recall
99.71%	99.06%	98.49%

the whole dataset.

In order to make the problem harder; we divided the data set into only 67% for training and 33% for testing. After generating the recreated training set, we chose 11% of its patterns randomly to train our SVM'ed- Kernel. The accuracy on the test set using our trained SVM'ed-Kernel turned out to be 99.71%. The precision was 99.06% while the recall was 98.49% as shown in Table 7.1 . We showed here that the SVM'ed-Kernel was able to generalize well even while training it with only 11% of the recreated training set. In addition, we illustrated the ability of the SVM'ed-Kernel to eliminate the burden of encoding the character attributes using an elegant definition of the similarity feature vector in a binary simple form.

Chapter 8

Scientific Paper Categorization

8.1 Problem Definition

In this problem, we are interested in categorizing scientific papers into either being related to the IEEE Computational Intelligence society or the IEEE Communications Society. We have solved this problem using the classical approach, by defining a set of key words and build the feature vector from their frequency of occurrence. For example, feature one could be the number of occurrences of the word Wireless.

To solve the problem using the SVM'ed Kernel, we have defined only two simple and effective features which are the number of common unigrams and the number of common bigrams. Figures 8.1 and 8.2 show a block diagram for the classical and The SVM'ed Kernel approaches respectively.

8.2 Results

We have extracted a data set from IEEE Xplore and divided it into 75% training set and 25% test set. For the classical approach, we have used 15 representative keywords, the best kernel performing was the RBF kernel. Table 8.1 shows the test set accuracy of the SVM'ed Kernel approach and the classical approach using the RBF kernel.

In this problem, if we had a new batch of papers served as an extension to the

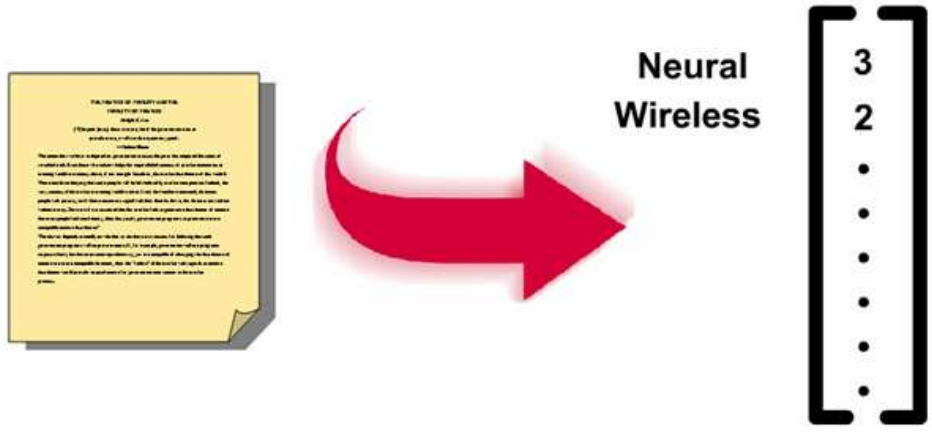

Figure 8.1: A block diagram showing a classical feature definition for the scientific paper categorization problem

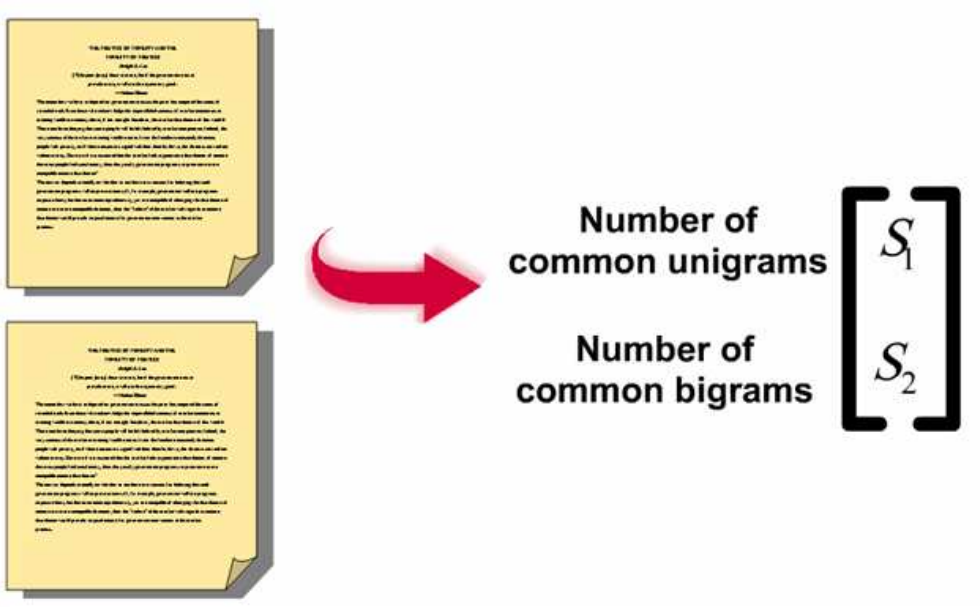

Figure 8.2: A block diagram showing the novel feature definition for the scientific paper categorization problem

Table 8.1: Testing accuracies for the scientific paper categorization problem

Model	RBF	SVM'ed Kernel
Accuracy	93.59%	96.15%

already existing training set. We will have to redesign the feature definition in the classical definition to reflect the necessity of key words newly introduced to the problem. For example, if a new model called support vector machines was developed in the field of Computational Intelligence, it will be important to redefine the features to include such new keyword.

On the Contrary, this is not the case for the SVM'ed Kernel approach. The two high level features proposed are generic and therefore there is no need for a redefinition. We will only have to add the new papers to the training set and retrain using the two high level features.

This shows how a high level similarity feature could be generic and release any dependence between the feature definition and specific training patterns.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this thesis, we have proposed the SVM'ed kernel function and its use in the classification problems. The novel kernel has introduced a new classification framework where a similarity feature vector is extracted for each pair. This allowed many details to be captured in a concise way as opposed to the original framework that requires each pattern to be converted to a feature vector representing it. Simulations show that the original defined features for classification problems could still be used together with novel highly effective features. Such novel features significantly lead to an increase in accuracy and eliminate the burden of feature encoding.

9.2 Future Work

In our future work, we intend to use the SVM'ed Kernel for unsupervised learning and to find the conditions under which the SVM'ed Kernel is admissible. Additionally, we plan to apply the new kernel to various real world problems in the fields of Bioinformatics and Computer Vision.

Chapter 10

Appendix - Supporting Figures and Illustrations

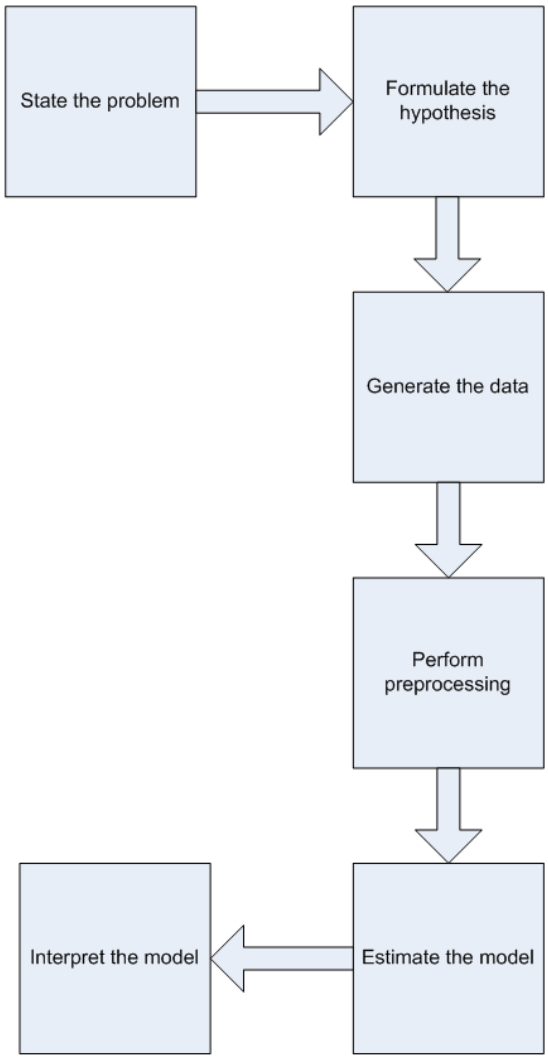

Figure 10.1: Learning Flow Chart

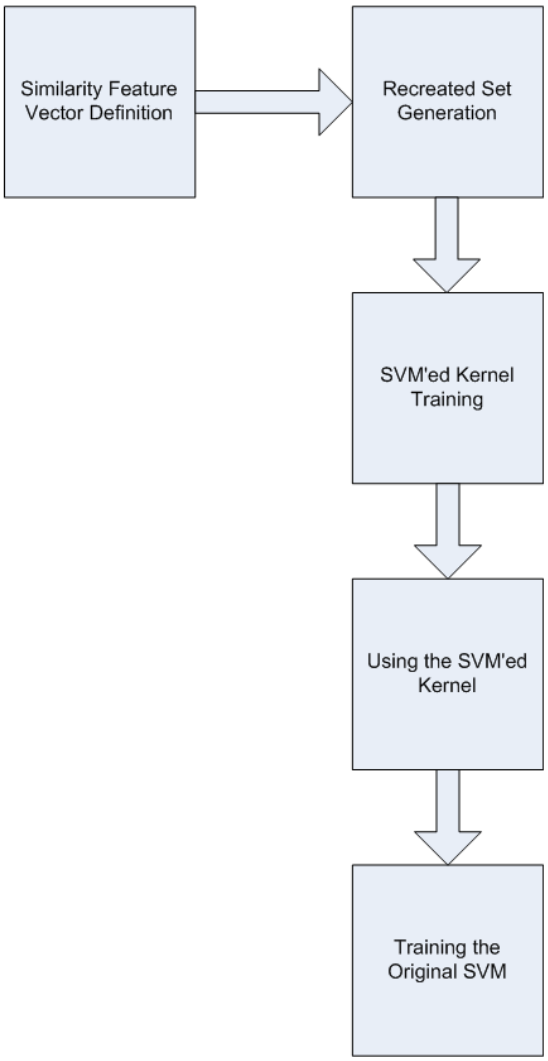

Figure 10.2: The SVM'ed Kernel Approach Steps

Bibliography

[1] Khaled S. Refaat: The Support Vector Machined Kernel. In: The IEEE Region 8 Eurocon. (2009)

[2] Scholkopf, B., Smola, A. J.: Learning with Kernels. The MIT Press Cambridge, Massachusetts London (2002)

[3] Thadani, K., Jayaraman, A.,and Sundararajan, V.: Evolutionary Selection of Kernels in Support Vector Machines. In: International Conference on Advanced Computing and Communication (2006)

[4] Kong, R., Zhang, B.: Autocorrelation Kernel Functions for Support Vector Machines. In: Third International Conference on Natural Computation (2007)

[5] N. Ye, R. Sun, Y. Liu and L. Cao.: Support Vector Machines with Orthogonal Chebyshev Kernel. In: The International Conference on Pattern Recognition (ICPR2006) (2006)

[6] S. Boughorbel, J. Tarel, and N. Boujemaa.: Conditionally Positive Definite Kernels for SVM Based Image Recognition.In: IEEE International Conference on Multimedia and Expo (ICME2005), pp.113-116. (2005)

[7] George, J., Rajeev, K.: SINC-CAUCHY Hybrid Wavelet Kernel for Support Vector Machines. In: IEEE Workshop on Machine Learning for Signal Processing. IEEE Press (2008)

[8] R. Cao and C. L. Tan: Line Primitive Extraction by Interpretation of Line Continuation, School of Computing, National University of Singapore.

[9] Wang, D., Yeung, D.,Eric, C.: Weighted Mahalanobis Distance Kernels for Support Vector Machines. In: IEEE Transaction on Neural Networks, vol. 18, pp. 1453–1462. IEEE Press (2007)

[10] Xu, J., Zhang, X.: Kernels Based on Weighted Levenshtein Distance. In: International Joint Conference on Neural Networks, pp. 3015–3018. IEEE Press, Budapest (2004)

[11] G. Siolas, and F. dAlche-Buc. Support Vector Machines Based on a Semantic Kernel for Text Categorization. In: International Joint Conference on Neural Networks (IJCNN2000), pp.205-209. (2000)

[12] Wu, X., Wang, J., Herbet, K.: A New Kernel Method for RNA Classification. In: Sixth IEEE International Symposium on BioInformatics and BioEngineering, pp. 201–208. Virginia (2006)

[13] Yan, C., Wang, Z., Gao, Q., Du, Y.: A Novel Kernel for Sequences Classification. In: IEEE International Conference on Natural Language Processing and Knowledge Engineering, pp. 769–773. Wuhan (2005)

[14] Cuturi, M., Vert, J.: A Mutual Information Kernel for Sequences. In: International Joint Conference on Neural Networks, pp. 1905–1910. IEEE Press, Budapest (2004)

[15] Refaat, K., Helmy, W., Ali, A., Abdelghany, M., Atiya, A.: A New Approach for Context-Independent Handwritten Offline Diagram Recognition using Support Vector Machines. In: International Joint Conference on Neural Networks, pp. 177-182. IEEE Press, Hong Kong (2008)

[16] SVMlight is an implementation of Support Vector Machines in C, by Thorsten Jaochims

[17] Boyce, J., Dobkin, D., Drysdale, R., Guibas, L.: Finding External Polygons. In: Annual Symposium on the Theory of Computing (1982)

[18] J. Joachims: The maximum-margin approach to learning text classifier: method, theory and algorithms, Ph.D. Department of Computer Science, University of Dortmund (2000)

[19] H. Kim and S. Park: Data Reduction in Support Vector Machines by a Kernalized Ionic Interaction Model. Proceedings of SIAM International Conference on Data Mining SDM2004 (2004)

[20] Valveny, E., Marti, E.: Deformable template matching within a Bayesian framework for hand-written graphic symbol recognition. In: Chhabra, K., Dori, D. (eds.) Graphics Recognition Recent Advances 2000. LNCS, vol. 1941, pp. 193–208. Springer, Heidelberg (2000)

[21] Notowidigo, M., Miller, C.: Offline sketch interpretation. In: AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural. Washington (2004)

[22] Ahmad, M., Park, J., Chang, M., Shim., Y., Choi, T.: Shape Registration based on Modified Chain Code. In: Zhou, X., Jahnichen, S., Xu, M., Cao, J. (eds.) Advanced Parallel Processing Technologies 2003. LNCS, vol. 2834, pp. 600–607. Springer, Heidelberg (2003)

[23] Microsoft software product for creating a wide variety of business and technical drawings

[24] M. J. Fonseca and J. A. Jorge: Using Fuzzy Logic to Recognize Geometric Shapes Interactively. Inthe 9th Int. Conference on Fuzzy Systems (FUZZ-IEEE 2000), Departamento de Engenharia Informatica, IST/UTL Av. Rovisco Pais, 1049-001 Lisboa, Portugal. (2000)

[25] Notowidigdo, M.: User-Directed Sketch Interpretation. MEng thesis, Massachusetts Institute of Technology (2004)

[26] Koby Crammer, Joseph Keshet, Yoram Singer. Kernel Design Using Boosting. In: Neural Information Processing Systems (2003)

[27] Huma Lodhi, John Shawe Tailor, Nello Christianini. Text Classification using String Kernels. In: Neural Information Processing Systems (2001)

[28] Konrad Rieck, Pavel Laskov, Soren Sonnenburg. Computation of Similarity Measures for Sequential Data using Generalized Suffix Trees. In: Neural Information Processing Systems (2007)

[29] Youngmin Cho and Lawrence K. Saul. Kernel Methods for Deep Learning. In: Neural Information Processing Systems (2009)

[30] Jing Bai, Xueying Zhang, Yueling Guo. Different inertia weight PSO algorithm optimizing SVM kernel parameters applied in a speech recognition system. In: ICMA 2009. International Conference on Mechatronics and Automation. (2009)

[31] Sahak R., Mansor W., Lee Yoot Khuan, Yassin A., Zabidi A., Rahman F.Y.A. Choice for a support vector machine kernel function for recognizing asphyxia from infant cries. In: IEEE Symposium on Industrial Electronics & Applications (2009)

[32] Xiangyang Mu, Weixin Gao, Nan Tang, Yatong Zhou.A novel Least Squares Support Vector Machine kernel for approximation. In: 7th World Congress on Intelligent Control and Automation (2008)

[33] Jianhua Xu, Xuegong Zhang. A multiclass kernel perceptron algorithm. In: International Conference on Neural Networks and Brain (2005)

[34] Nemmour, H., Chibani, Y. New Jaccard-Distance Based Support Vector Machine Kernel for Handwritten Digit Recognition. In: 3rd International Conference on Information and Communication Technologies: From Theory to Applications (2008)

[35] Campbell, W.M., Sturim, D.E., Reynolds, D.A., Solomonoff, A.. SVM Based Speaker Verification using a GMM Supervector Kernel and NAP Variability Compensation. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2006)

[36] Ayat, N.E., Cheriet, M., Remaki, L., Suen, C.Y.. KMOD - a new support vector machine kernel with moderate decreasing for pattern recognition. Application to digit image recognition. In: Sixth International Conference on Document Analysis and Recognition (2001)

[37] Jalam, R., Teytaud, O. Kernel-based text categorisation. In: International Joint Conference on Neural Networks (2001). IEEE Conferences

[38] Huazhu Song, Zichun Ding, Cuicui Guo, Zhe Li, Hongxia Xia. Research on Combination Kernel Function of Support Vector Machine. In: International Conference on Computer Science and Software Engineering (2008)

[39] Cai-Xia Deng, Li-Xiang Xu, Zuo-Xian Fu. The Beat-wave signal regression based on least squares reproducing kernel support vector machine. In: International Conference on Machine Learning and Cybernetics (2008)

[40] Song Haiying, Gui Weihua, Yang Chunhua. Reduced Least Squares Support Vector Based on Kernel Partial Least Squares and Its Application Research. In: Control Conference, 2007. CCC 2007. Chinese (2007)

[41] Zhang Qizhong. Gene Selection and Classification Using Non-linear Kernel Support Vector Machines Based on Gene Expression Data. In: IEEE/ICME International Conference on Complex Medical Engineering (2007)

[42] Jing Bai, Yueling Guo. Speech Recognition Method Based on Linear Descending Inertia Weight PSO Algorithm Optimizing SVM Kernel Parameters. In: Fifth International Conference on Natural Computation (2009)

[43] Bo Yang, Ying-yong Bu. Multiple Kernel LSSVM in Empirical Kernel Mapping Space. In: IITA International Conference on Control, Automation and Systems Engineering (2009)

[44] Yue Jiang, Changjie Tang, Chuan Li, Shengzhi Li, Shangyu Ye, Taiyong Li, Haichun Zheng. Automatic SVM Kernel Function Construction Based on Gene

Expression Programming. In: International Conference on Computer Science and Software Engineering (2008)

[45] Hu-Sheng Guo, Wen-Jian Wang, Chang-Qian Men. A novel learning model-Kernel Granular Support Vector Machine. In: International Conference on Machine Learning and Cybernetics (2009)

[46] Guo-Feng Pan, Ping He, Ya-Tong Zhou, Jian-Hua Li. Constructing a wavelet-based RKHS and its associated scaling kernel for support vector approximation. In: International Conference on Wavelet Analysis and Pattern Recognition (2007)

[47] Li Cong-Cong, Guo Ai-ling, Li Dan. Combined Kernel SVM and Its Application on Network Security Risk Evaluation. In: International Symposium on Intelligent Information Technology Application Workshops (2008)

[48] Yan-Ling Lu, Lei Li, Meng-Meng Zhou, Guo-Liang Tian. A new fuzzy support vector machine based on mixed kernel function. In: International Conference on Machine Learning and Cybernetics (2009)

[49] Chaudhuri, A., De, K., Chatterjee, D.. A Comparative Study of Kernels for the Multi-class Support Vector Machine. In: Fourth International Conference on Natural Computation (2008)

[50] Meng Yafeng, Ren Mingqiu, Cai Jinyan, Han Chunhui. Research on Radar Emitters Classification with Fuzzy Support Vector Machines. In: International Forum on Information Technology and Applications (2009)

[51] Xiangtao Wang, Yan Feng. New Method Based on Support Vector Machine in Classification for Hyperspectral Data. In: International Symposium on Computational Intelligence and Design (2008)

[52] Kostka, P.S., Tkacz, E.J. Feature extraction for improving the support vector machine biomedical data classifier performance. In: International Conference on Information Technology and Applications in Biomedicine (2008)

[53] Chuangxin He, Yanming Li, Yixiang Huang, Chengliang Liu, Shengwei Fei. Relevance Vector Machine Based Gear Fault Detection. In: Chinese Conference on Pattern Recognition (2009)

[54] Haibin Ling, Soatto, S.. Proximity Distribution Kernels for Geometric Context in Category Recognition. In: IEEE 11th International Conference on Computer Vision (2007).

[55] Jianhua Xu, Xuegong Zhang, Yanda Li. Sparse training procedure for kernel neuron. Proceedings of the 2003 International Conference on Neural Networks and Signal Processing (2003)

[56] Weiwei Xing; Weibin Liu; Baozong Yuan. Superquadric-based geons recognition utilizing support vector machines. In: International Conference on Signal Processing (2004)

[57] Zhang Haihong, Guan Cuntai. A Kernel-based Signal Localization Method for NIRS Brain-computer Interfaces. In: 18th International Conference on Pattern Recognition (2006)

[58] Mori, T.; Shimosaka, M.; Harada, T.; Sato, T.. Informative motion extractor for action recognition with kernel feature alignment In: Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (2004).

[59] Sun, B; Li, J; Wu, D; Zhang, X; Li, W. Kernel Discriminant Learning for Ordinal Regression. In: IEEE Transactions on Knowledge and Data Engineering (2009)

[60] Guo, B.; Gunn, S.; Damper, B.; Nelson, J. Hyperspectral image fusion using spectrally weighted kernels. In: 8th International Conference on Information Fusion (2005)

[61] Zhang, M.; Fu, L.; Li, H.. Harmonic Retrieval Based on LS-SVM with Wavelet Kernel. In: 8th International Conference on Signal Processing (2006)

[62] Ben-Hur, A.; Horn, D.; Siegelmann, H.T.; Vapnik, V.. A support vector clustering method. In: 15th International Conference on Pattern Recognition (2000).

[63] Jun-Bao Li, Shu-Chuan Chu, Jeng-Shyang Pan, Jiun-Huei Ho. A Novel Matrix Norm Based Gaussian Kernel for Feature Extraction of Images. In: International Conference onIntelligent Information Hiding and Multimedia Signal Processing (2006)

[64] Lunbo Li, Jun Li, Jianhong Sun. Traffic Sign Classification Based on Support Vector Machines and Tchebichef Moments. In: International Conference on Computational Intelligence and Software Engineering (2009)

[65] Jianping Li, Zhenyu Chen, Weixuan Xu. Bounded Support Vector Machines, Semidefinite. In: Sixth IEEE International Conference on Data Mining Workshops (2006)

[66] Zhu Lingyun, Cao Changxiu, Wu Wei, Xu Xiaoling. A novel approach based on support vector machine to forecasting the quality of friction welding. Proceedings of the 4th World Congress on Intelligent Control and Automation (2002)

[67] Junxian Li, Limin Shen, Shuo Yang. A Novel Radar Target Recognition Algorithm Based on SVM. In: International Symposium on Intelligent Information Technology Application Workshops (2008)

[68] Qi-Song Chen, Xin Zhang, Shi-Huan Xiong, Xiao-Wei Chen. The combining kernel PCA with PSO-SVM for chaotic time series prediction model. International Conference onMachine Learning and Cybernetics (2009)

[69] Gubbi, J.; Shilton, A.; Palaniswami, M.; Parker, M.. Real Value Solvent Accessibility Prediction using Adaptive Support Vector Regression. In: IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology (2007)

[70] Shimosaka, M.; Mori, T.; Harada, T.; Sato, T. Action recognition based on kernel machine encoding qualitative prior knowledge. In: IEEE International Conference on Systems, Man and Cybernetics (2004)

[71] Fukuda, S.; Hirosawa, H. Support vector machine classification of land cover: application to polarimetric SAR data. In: IEEE 2001 International Geoscience and Remote Sensing Symposium (2001)

[72] Salimi, F.; Sadeghi, M.T. Composite Kernels for Fusing Colour Information in Face Verification Systems. In: International Conference on Digital Image Processing (2009)

[73] Junping Zhang; Ye Zhang; Tingxian Zhou. Classification of hyperspectral data using support vector machine. In: 2001 International Conference on Image Processing (2001)

[74] Nilufar, Sharmin; Ray, Nilanjan; Zhang, Hong. Optimum kernel function design from scale space features for object detection. 16th IEEE International Conference on Image Processing (ICIP) (2009)

[75] Williamson, R.C.; Smola, A.J.; Scholkopf, B.. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators IEEE Transactions on Information Theory (2001)

[76] Guoyun Zhang, Jing Zhang. Face recognition using multi-scale PCA and Support Vector Machine. 7th World Congress on Intelligent Control and Automation (2008)

[77] Chin-Chun Chang. Deformable shape finding with models based on kernel methods. IEEE Transactions on Image Processing,

[78] Y an Wang, Xueyan Liu, Yujuan Xing, Ming Li. A Novel Reduction Method for Text-Independent Speaker Identification. Fourth International Conference on Natural Computation (2008)

[79] Rojo-Alvarez, J.L.; Figuera-Pozuelo, C..; Martinez-Cruz, C.E.; Camps-Valls, G..; Alonso-Atienza, F..; Martinez-Ramon, M. Nonuniform Interpolation of Noisy Signals Using Support Vector Machines In: IEEE Transactions on Signal Processing (2007)

[80] Madzarov, G.; Gjorgjevikj, D. Multi-class classification using support vector machines in decision tree architecture In: IEEE EUROCON (2009)

[81] Campbell, W.M.; Campbell, J.P.; Gleason, T.P.; Reynolds, D.A.; Wade Shen. Speaker Verification Using Support Vector Machines and High-Level Features. In: IEEE Transactions on Audio, Speech, and Language Processing (2007)

[82] Ma Guorui; Sui Haigang; Li Pingxiang; Qin Qianqing. A Kernel Change Detection Algorithm in Remote Sense Imagery. In: IEEE International Conference on Geoscience and Remote Sensing Symposium (2006).

[83] Xiao-Zhang Liu, Wen-Sheng Chen, Yuen, P.C., Guo-Can Feng. Learning Kernel in Kernel-Based LDA for Face Recognition Under Illumination Variations. In: IEEE Signal Processing Letters (2009).

[84] Dit-Yan Yeung, Hong Chang. A Kernel Approach for Semisupervised Metric Learning. In: IEEE Transactions on Neural Networks (2007)

[85] Cortes, C.; Mohri, M.; Rostamizadeh, A.. Learning sequence kernels. In: IEEE Workshop on Machine Learning for Signal Processing, (2008)

[86] Seung-Jean Kim; Zymnis, A.; Magnani, A.; Kwangmoo Koh; Boyd, S.. Learning the kernel via convex optimization. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2008).

[87] Bo Yang; Yingyong Bu. Multiple Kernel Learning Using Regularized Ho-Kashyap Classifier in Empirical Kernel Mapping Space. In: Fifth International Conference on Natural Computation (2009)

ملخص الرسالة

نقترح ما يسمي بالنواة المتجهية الداعمة و استخدامها في التصنيف. هذه النواة هي في حد ذاتها مصنف آلة متجهات داعمة يتم تعلمه من البيانات إحصائيا. النواة الجديدة يمكنها تغيير المنهجية الكلاسيكية التي تتطلب تحديد متجه سمات لكل نمط واحد حيث لا تتطلب تلك النواة إلا تحديد السمات التي تمثل التشابه بين نمطين مما يسمح للكثير من التفاصيل ان تظهر بطريقة موجزة. تم اختبار النواة الجديدة في عدة من مشكلات العالم الحقيقية و البحثية و قد أظهرت نتائج واعدة للغاية. إن هذه النواة تفتح الباب لتعريف سمات جديدة في مختلف مشاكل التعلم حيث يمكن أن يصاغ التشابه بين الأنماط بشكل أنسب.

.

النواة المتجهية الداعمة: نحو منهجية جديدة للتصنيف

إعداد

خالد سعيد سعد زغلول علي رفعت
رسالة مقدمة إلى كلية الهندسة، جامعة القاهرة

كجزء من متطلبات الحصول على درجة الماجستير
في هندسة الحاسبات

يعتمد من لجنة الممتحنين:

أستاذ دكتور أمير فؤاد سوريال عطية	مشرف رئيسي
أستاذ دكتور نيفين محمود درويش	عضو
أستاذ دكتور علي علي فهمي	عضو

كلية الهندسة ، جامعة القاهرة

الجيزة ، جمهورية مصر العربية

مايو 2010

النواة المتجهية الداعمة: نحو منهجية جديدة للتصنيف

إعداد

خالد سعيد سعد زغلول علي رفعت

رسالة مقدمة إلى كلية الهندسة، جامعة القاهرة

كجزء من متطلبات الحصول على درجة الماجستير

في هندسة الحاسبات

تحت إشراف

أستاذ دكتور

أمير فؤاد سوريال عطية

جامعة القاهرة

كلية الهندسة ، جامعة القاهرة

الجيزة ، جمهورية مصر العربية

مايو 2010

النواة المتجهية الداعمة: نحو منهجية جديدة للتصنيف

إعداد

خالد سعيد سعد زغلول علي رفعت

رسالة مقدمة إلى كلية الهندسة، جامعة القاهرة

كجزء من متطلبات الحصول على درجة الماجستير

في هندسة الحاسبات

كلية الهندسة ، جامعة القاهرة

الجيزة ، جمهورية مصر العربية

مايو 2010