Meta Learning on String Kernel SVMs for String Categorization

Nuwan Amila Gunasekara

A thesis submitted to

Auckland University Of Technology

in fulfillment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

15th July 2010

Knowledge Engineering and Discovery Research Institute (KEDRI)

Primary Supervisor: Dr. Paul S. Pang

Secondary Supervisor: Prof. Nik Kasabov

Additional Supervisor: Dr. Tao Ban

Contents

List of Figures

List of Tables

List of Algorithms

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Nuwan Amila Gunasekara.

Acknowledgment

The thesis would not have been a success without many people. I wish to express my sincere gratitude to my supervisors: Prof. Nik Kasabov, Dr. Paul S. Pang and Dr Tao Ban, who have abundantly provided me with invaluable assistance and guidance. Also, many thanks go to Joyce D'Mello, the Manager of KEDRI, who has always being supportive to me.

All the KEDRI members have supported me with their knowledge and assistance in difficult moments of the my research. I wish to thank them all for their support.

Next, my deepest gratitude goes to my team mates: Lei Song, Huili Song (Alex), Gary Chen and Liu Fan (Richard) at Pattern Recognition Center, who have helped me immensely to improve my research work by their shared knowledge and constructive criticism.

I would also like to convey my thanks to NICT, Japan, for providing me with a scholarship and opportunity to work on their project. Many thanks go to Auckland University of Technology for providing a good study environment and facilities.

Also, I would like to thank Heather Moodie for proofreading my thesis and suggesting necessary corrections.

Last, but not the least, my love and gratitude go to my parents for their understanding, support and endless love, through the course of my life.

Abstract

Even though Support Vector Machines (SVMs) are capable of identifying patterns in high dimensional spaces that are presented by kernels without a computational decrease, their performance is determined by two main factors: SVM cost parameter and kernel parameters. It is empirically proven that the optimum parameter combination for both SVM cost and kernel yield high pattern recognition accuracies in SVMs. Various methods are discussed in the literature to optimize these SVM parameters, namely trial and error method, grid optimization, leave-one-out cross validation, generalization error estimation using gradient descent, and evolutionary algorithms. However, these optimization methods have downfalls: the trial and error method is considered to be imprecise and unreliable (Zhang & He, 2010), the grid parameter search requires complex computation (Friedrichs & Igel, 2005), leave-one-out cross validation is considered to be computationally expensive, and the gradient descent method can fall into a local 'minima' (Zhang & He, 2010). Also, these methods mainly optimize numeric kernels. However, when optimizing string kernel SVMs in string classification, researchers are confronted with two main obstacles. Firstly, there is little or no literature pertaining to string kernel SVM optimization at the moment. Secondly, according to the experiments: Rieck and Laskov (2007), Lodhi, Saunders, Shawe-Taylor, Cristianini, and Watkins (2002), Sharma, Girolami, and Sventek (2007) and S.Sonnenburg (2006), string dataset characteristics influence the optimum parameter combination of string kernel SVMs. The current string kernel SVM optimization methods do not take string dataset characteristics into account.

Baring this in mind, the thesis initially attempts to identify a mechanism to extract string dataset characteristics from a string dataset. It then, attempts to derive a string kernel SVM optimization method which uses these extracted string dataset characteristics. The initial objective is achieved by explaining that a sting dataset

is presentable in *token-frequency* space (where each dimension is represented by a token) and then computing *string meta-features* by the points of this token-frequency space. Now for a machine learning algorithm, a *meta model* is trained using computed string meta-features for each dataset in a string dataset pool, learning algorithm parameters, and accuracy information. This trained meta model helps to predict string classification accuracy for a new string dataset by computing relevant string meta-features. This principle is employed to optimize string kernel SVMs in the proposed meta learning algorithm (this fulfills the second objective of the thesis) where it only computes the relevant string meta-features to yield optimum parameter combination for the machine learning algorithm on the new string dataset.

In the experiments, three string kernel SVMs, *edit-distance* SVM, *bag-of-words* SVM and *n-gram* SVM, were optimized using the proposed algorithm on four string datasets: spam, Reuters-21578, Network Application Detection and e-News Categorization. The experiment results revealed that the algorithm was able to produce parameter combinations which yield good string classification accuracies for string kernel SVMs on most string datasets. It is also revealed that some string kernel SVMs may not be suitable on certain string datasets.

As future work, one could research for more string meta-features that help to employ meta learning on string classification more effectively. Also, one could introduce upper and lower bounds to the proposed algorithm, which help the predicted string classification accuracy to be within the range (0,100). Further experiments on more string datasets will help to identify the robustness of the algorithm.

Chapter 1

Introduction

Support Vector Machines (SVMs) are a set of supervised learning techniques, which use statistical learning principles, kernel mapping, and optimization techniques for classification and regression. SVMs were originally introduced by Boser, Guyon, and Vapnik (1992). In its simplest form, SVM learns a separating hyperplane which maximizes the distance between the hyperplane and its closest point by solving a convex quadratic optimization problem. The optimization problem has several interesting statistical properties which make SVMs suitable for generalization. Moreover, kernel mapping allows SVMs to work with non linearly separable data, by mapping them into high dimensional feature space.

As Shawe-Taylor and Cristianini (2004) elucidate, there are seven main kernel families: kernels in closed form (i.e. gaussian and polynomial), ANOVA kernels, kernels from graphs, diffusion kernels on graph nodes, kernels on sets, kernels on real numbers and randomized kernels. Also, there are application specific kernels such as string kernels (i.e. bag-of-words kernel, edit-distance kernel, n-gram kernel, subsequence, etc.) which are used for specific classification tasks like document classification or network anomaly detection (Lodhi et al., 2002; Rieck & Laskov, 2007). Furthermore, one can derive a novel kernel function that satisfies the 'finitely positive semi-definite property' (Shawe-Taylor & Cristianini, 2004). Even though there are bulk of viable kernel functions, each has its own unique way of deriving data into feature space, allowing the kernel selection to have considerable influence on the performance of a pattern recognition task (Chapelle, Vapnik, Bousquet, & Mukherjee, 2002; Frohlich & Zell, 2005).

The effectiveness of SVM is heavily dependent upon three main factors: kernel selection, SVM cost parameter, and kernel parameters (Chapelle et al., 2002; Frohlich & Zell, 2005). Apparently, as diverse sets of kernel functions are available, identifying the most suitable one for a given pattern recognition task is quite challenging, where the researcher spends considerable time on kernel identification. On the other hand, for a suitable kernel function, the performance is again influenced by two factors: SVM cost (C) parameter and kernel parameters. As Rieck and Laskov (2007) and Burges (1998) explain different SVM cost values yield different classification accuracies. Also, according to Zhang and He (2010) and Friedrichs and Igel (2005), multiple kernel parameters affect the performance of SVMs pattern recognition ability. For an example, parameters like gamma (γ) and coefficient in polynomial kernel, and parameter sigma (σ) in gaussian kernel, largely influence the pattern recognition ability of SVM (Chapelle et al., 2002; Eitrich & Lang, 2006). Also, for string kernel parameters like substring length in n-gram kernel and subsequence size in fixed length subsequence kernel affects the performance of SVM (Lodhi et al., 2002; Rieck & Laskov, 2007). Hence, the initial problem of kernel identification, and then SVM kernel parameter optimization, can be as challenging as the initial pattern recognition task, due to the higher dimensionality of the parameter space (Eitrich & Lang, 2006).

Various optimization techniques are being used in SVM kernel parameter optimization, namely trial and error method, grid optimization, leave-one out cross validation, generalization error estimation using gradient descent and evolutionary algorithms. Each of these optimization methods has its own disadvantages: the trial and error method is considered to be imprecise and unreliable (Zhang & He, 2010), the grid parameter search requires complex computation (Friedrichs & Igel, 2005), leave-one out cross validation is considered to be computationally expensive and the gradient descent method can fall into a local 'minima' (Zhang & He, 2010). The above mentioned SVM optimization methods mainly optimize numeric kernels. But, for string kernel SVM optimization, researchers are confronted with two main obstacles. Firstly, there is very little or no literature pertaining to string kernel SVM optimization at the moment. Secondly, as experiments: Rieck and Laskov (2007), Lodhi et al. (2002), Sharma et al. (2007) and S.Sonnenburg (2006) explain, string dataset characteristics also influence string kernel SVM optimization. This allows one to include string dataset characteristics into a potential string kernel SVM optimization

method. The thesis explains a novel string kernel SVM optimization method taking string dataset characteristics, meta learning techniques and regression methods into account.

1.1 Motivation

The research work addressed in this thesis is originally inspired by works of Lam and Lai (2001) and Furdík, Paralič, and Tutoky (2008), where meta learning is used to select algorithms for each text category in text categorization. In their approach, the *text meta-features* which are extracted from a text dataset along with model performance information (classification accuracy and root mean squared error) are used to train a prediction model that predicts the performance of each algorithm on each text category. The algorithm with the highest predicted performance on a particular text category is chosen to perform the actual text categorization on that category. The *text meta-features* are calculated using *terms* and their frequencies in a text dataset. The approach helps to acquire meta knowledge about a novel text dataset by calculating its relevant text meta-features. Later, this text meta knowledge is used for model selection in text classification.

Similarly, the meta knowledge about a string dataset is acquirable (or the *string meta-features* are computable), if the string dataset is presented in the form of terms and their frequencies, so that, the acquired string meta knowledge (from string meta-features) can be used to select models in string classification.

1.2 Objectives and Approach

1.2.1 Research Questions

The research questions addressed in this thesis are:

Can a string dataset be represented in such a way like *terms* and their frequencies as in Lam and Lai (2001) and Furdík et al. (2008)?

If one is able to represent a string dataset as terms and their frequencies, then using the calculated sting meta-features via terms and their frequencies,

How can we optimize string kernel SVMs using meta learning?

Here, the initial step is to identify a mechanism which transforms a string dataset to a space like term-frequency space. Next, the sting meta-features are calculated by points in this new space, so that the sting meta-features can be used to train a meta model which optimize string kernel SVMs.

1.2.2 Approach

Considering the idea mentioned in section 1.2.1 for string dataset representation, section 4.2 explains a novel method of representing a string dataset in the *token-frequency* space. Now string meta-features are computed by the points in this token-frequency space. Later, section 4.3 explains the principle of meta learning for string classification, where a *meta model* is trained for a machine learning algorithm using computed string meta-features for each dataset in a string dataset pool, learning algorithm parameters, and accuracy information. This principle is used to derive a novel string kernel SVM optimization algorithm in section 4.3.2, where string kernel SVMs are optimized using computed string meta-features.

1.3 Thesis Structure

The thesis is structured as follows:

Chapter 2 discusses the main concepts of SVM by explaining SVM, kernel trick, string kernels and SVM parameter optimization.

Chapter 3 explains the principles of meta learning. It also discusses the application of meta learning for text categorization by explaining text meta-features. The chapter clearly defines text data and string data which brings about the motivation to use meta learning for string classification.

Chapter 4 initially, explains the mechanism to present a string dataset into the *token-frequency* space. It then elucidates sting meta-features, which are calculated by the points in this token-frequency space. The chapter then discusses the principle of using meta learning for string classification. Based on this principle, the chapter explains a novel string kernel SVM optimization algorithm via meta learning.

Chapter 5 discusses the experimental results for optimizing three string kernel SVMs (*edit-distance* SVM, *bag-of-words* SVM and *n-gram* SVM) using the proposed meta learning algorithm for string kernel SVM optimization, on four string datasets. It also explains the experimental setup, evaluation criteria and dataset description.

Chapter 6 concludes the thesis by discussing contributions, limitations and future work.

Chapter 2

String Kernel SVM

2.1 Introduction

As mentioned in previous chapter, the main objective of the research is to optimize string kernel SVMs. Hence, this chapter focuses on explaining concepts related to SVMs such as, SVMs, kernel trick and SVM optimization. Section 2.2 discuses the principles of SVMs and kernel trick. The section also explains three string kernels, namely edit-distance, bag-of-words and n-gram. Furthermore, section 2.3 discusses the issue of SVM optimization along with currently available SVM optimization strategies specially for numeric kernels, followed by string kernel SVM optimization in section 2.4.

2.2 SVMs and Kernel Trick

To understand the principle of SVMs, we consider a simple binary classification situation. For a given training data set D $((x_i, y_i), ..., (x_n, y_n))$ in input space X ($X \in \mathbb{R}^d$) having class label y ($\{-1, +1\}$), there exists an element ω where $\|\omega\|_2 = 1$ and b ($b \in \mathbb{R}$) such that,

$$\langle \omega, x_i \rangle + b > 0 \text{ for all } i \text{ with } y_i = +1$$

$$\langle \omega, x_i \rangle + b < 0 \text{ for all } i \text{ with } y_i = -1$$

According to Boser et al. (1992), a solution to the above can be found by solving the below optimization problem:

$$\begin{aligned} &\text{minimize} && \langle \omega, \omega \rangle \text{ over } \omega \in \mathbb{R}^d,\ b \in \mathbb{R} \\ &\text{subject to} && y_i(\langle \omega, x_i \rangle + b) \geq 1,\ i = 1, ..., n \end{aligned} \tag{2.1}$$

Although the solution of (2.1) optimization is geometrically compelling, it has two shortcomings by which it avoids generalization:

1. The linear form of the decision function is not ideal for situations where the training data set is not linearly separable.

2. The current form of the decision function tends to *overfit* in most instances, hence a mechanism is required to misclassify some data to avoid *overfitting*.

To resolve the first issue, the input data $(x_i, ..., x_n)$ is mapped into a possibly-infinite-dimensional Hilbert space H_o (feature space) using non liner mapping $\Phi : X \rightarrow H_o$, and then the *generalized portrait algorithm* (Vapnik, 1963) is applied to data $((\Phi(x_i), y_i), ..., (\Phi(x_n), y_n))$ in this new space (H_o). The second issue is addressed in *soft margin support vector machines* (Cortes & Vapnik, 1995), by introducing slack variables $(\xi_i, ..., \xi_n)$ to the objective function. The idea of feature mapping and slack variables ultimately leads (2.1) to a quadratic optimization problem as,

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\langle \omega, \omega \rangle + C \sum_{i=1}^{n} \xi_i \text{ for } \omega \in H_o,\ b \in \mathbb{R},\ \xi \in \mathbb{R}^n \\ &\text{subject to} && y_i(\langle \omega, \Phi(x_i) \rangle + b) \geq 1 - \xi_i,\ i = 1, ..., n \\ & && \xi_i \geq 0,\ i = 1, ..., n \end{aligned} \tag{2.2}$$

Here, C is a penalty parameter which is often determined through cross validation tests. One can rewrite the first constrain in (2.2) as $\xi_i \geq 1 - y_i(\langle \omega, \Phi(x_i) \rangle + b)$ and combine it with second constraint $(\xi_i \geq 0)$ to get,

$$\xi_i \geq \max\{0, 1 - y_i(\langle \omega, \Phi(x_i) \rangle + b)\} = L(y_i, \langle \omega, \Phi(x_i) \rangle + b), \tag{2.3}$$

where L is the *hinge loss*. One drawback of this new optimization problem (2.2) is that it needs to be solved in a high or infinite dimensional Hilbert space H_o. But, in

practice, the *Lagrange approach* is used to compute the corresponding dual program for *hinge loss*,

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle \text{ over } \alpha \in [0, c]^n \\ &\text{subject to} \quad \sum_{i=1}^{n} y_i \alpha_i = 0, \end{aligned} \tag{2.4}$$

If $(\alpha_i^*, ..., \alpha_n^*)$ is the solution to (2.4), then the solution (ω_D^*, b_D^*) to the optimization problem (2.2) can be computed as below,

$$\omega_D^* = \sum_{i=1}^{n} y_i \alpha_i^* \Phi(x_i)$$

and

$$b_D^* = y_j - \sum_{i=1}^{n} y_i \alpha_i^* \langle \Phi(x_i), \Phi(x_j) \rangle \, ,$$

where j is an index $(0 < \alpha_j^* < C)$. One can note that ω_D^* only depends on samples x_i where $\alpha_i^* \neq 0$. This means that hyperplane described by (ω_D^*, b_D^*) is only supported by those $\Phi(x_i)$s, where $\alpha_i^* \neq 0$. The relevant data points (x_i, y_i) are called 'support vectors' (Cortes & Vapnik, 1995). The decision function $f_{\omega_D^*, b_D^*} : X \to \mathbb{R}$ is written as,

$$f_{\omega_D^*, b_D*}(x) = \langle \omega_D^*, \Phi(x) \rangle + b_D^* = \sum_{i=1}^{n} y_i \alpha_i^* \langle \Phi(x_i), \Phi(x) \rangle + b_D^* \;\; x \in X \tag{2.5}$$

Interestingly, in both the dual optimization problem (2.4) and the decision function (2.5) only the inner product of Φ with itself occur. Hence, instead of computing the feature map (Φ), it is sufficient to know about a function $\langle \Phi(.), \Phi(.) \rangle : X \times X \to \mathbb{R}$. As Cortes and Vapnik (1995) explain, there are instances where we can compute this function $(\langle \Phi(.), \Phi(.))$ without knowing about the feature map (Φ) itself. Especially for kernels functions $k : X \times X \to \mathbb{R}$, there exists a feature map as below,

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle \;\; x, z \in X \tag{2.6}$$

The method of directly using kernels instead of computing the feature map, works well for algorithms which require the inner product of the feature map but not the feature map itself. This approach is known as 'kernel trick', it was first introduced by Aizerman, Braverman, and Rozoner (1964).

2.2.1 Kernel Trick

As the idea of kernel is raised in previous subsection, this subsection discuses the functionality of kernel and kernel matrix in detail. According to Shawe-Taylor and Cristianini (2004), the kernel is a function for all $x, z \in X$ that satisfies:

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle,$$

where, Φ is a mapping from X to a feature (inner product) space F. Also, the *gram matrix* G $(l \times l)$ for a set of vectors $S = \{\mathbf{x}_1, ..., \mathbf{x}_l\}$, is defined as:

$$G_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle.$$

Now using the idea of kernel function k with feature map Φ, the entries for the gram matrix are rewritten as:

$$G_{ij} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j) \text{ for } i, j = 1, ..., l.$$

This form of gram matrix is referred to as the 'kernel matrix' in literature (Shawe-Taylor & Cristianini, 2004). Furthermore, these kernel matrices are *positive semi-definite* for all training data sets, which is called the *finitely positive semi-definite property* of the kernel function (Shawe-Taylor & Cristianini, 2004). This positive semi-definite property of kernel matrix enables it be manipulated without concerning about the underlying feature map. This allows the kernel function to be redefined as follows. The kernel is a function,

$$k : X \times X \to \mathbb{R},$$

from a continuous or finite domain X, through pairs:

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle \quad x, z \in X,$$

via feature map Φ into *Hilbert space* F, if and only if it satisfies the finitely positive semi-definite property.

This definition of kernel function does not require the X to be a vector space, where any input space is viable as long as the relevant kernel matrix is positive semi-definite. This means one can use kernel functions like string kernels, where their input spaces are not vector spaces, but the relevant kernel matrices are positive semi-definite.

2.2.2 String Kernels

SVMs have shown success in numerical pattern recognition with the help of the kernel functionality (Chow, Zhong, Blackmon, Stolz, & Dowell, 2008; Steinwart & Christmann, 2008). Which has led the researchers to focus on using SVMs with string kernels for string classification (Lodhi et al., 2002; Rieck & Laskov, 2007). As Shawe-Taylor and Cristianini (2004) and Lodhi et al. (2002) explain, apart from popularly used string kernels like bag-of-words and edit-distance, one can also use string kernels like n-gram kernel or fixed length subsequence kernel to classify strings. Results from previous experiments by Shawe-Taylor and Cristianini (2004), Lodhi et al. (2002) and Rieck and Laskov (2007) show that an SVM with string kernel functionality is able to recognize string patterns much more efficiently than other methods due to its ability to handle high dimensional data like string data without a performance decrease. This has made the SVM with string kernels as an ideal candidate for DNA prediction, document classification, language recognition, image recognition and network anomaly detection tasks. The following subsections explain three widely used string kernels, namely edit-distance, bag-of-words and n-gram string kernels.

Levenshtein Distance (Edit-Distance)

The Levenshtein (or edit) distance computes the difference between two strings. The computed difference refers to the number of insertions, substitutions and deletions required to transform a string s to string t. Initially, the function r is defined as below:

$$r(a,b) = \begin{cases} 0 & \textit{if } a{=}b \\ 1 & \textit{otherwise.} \end{cases}$$

where, a and b are two characters.

Now, the $(n+1),(m+1)^{th}$ item of the matrix M $(n+1) \times (m+1)$ furnishes the Levenshtein distance $L(s,t)$ between string s and string t. Here, M is computed in a recursive manner by, initially setting $M(i,0) = i$ for all $i = 1, 2, ..., n$ and $M(0,j) = j$ for all $j = 1, 2, ..., m$ and then, computing all $M(i,j)$ as below:

$$M(i,j) = min(M(i-1,j)+1, M(i,j-1)+1, M(i-1,j-1)+r(s(i),t(j))).$$

Here, n is the length of string s and m is the length of t. Also, $s(i)$ is the i^{th} character of s and $t(j)$ is the j^{th} character of t. Furthermore, in order to get positive values, $L(s,t)$, $e^{-\lambda_L M(i,j)}$ is used, where $\lambda_L \in (0,1)$ is a decay factor. The computational complexity of Levenshtein distance is $O(|s||t|)$.

Bag-of-Words Kernel

In document categorization, a collection of *documents* is called a 'corpus', which consists of a set of predefined *terms* and is identified as a *dictionary*. A term or synonymously a *word* in the dictionary is any sequence of letters separated by punctuation or spaces. On the other hand, a *bag*, is defined as a set which allows repeated items. This definition of bag helps one to view a document as a *bag of terms* or *bag of words*. This allows a document to be presented as a vector where each dimension is associated with a term in the dictionary. This representation (Φ) is given as:

$$\Phi : d \longmapsto \phi(d) = (tf(t_1,d), tf(t_2,d), ..., tf(t_N,d)) \in \mathbb{R}^N,$$

here $tf(t_i,d)$ is the frequency of the t_i^{th} *term* in document d. Also, N is the space dimensionality and the size of the dictionary (Shawe-Taylor & Cristianini, 2004). Now, one can define a function k in this document space to compare the similarity between two documents d_1 and d_2:

$$k(d_1,d_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f(t_i,t_j)^j,$$

where, d_1 has n_1 *terms* and d_2 has n_2 *terms*. Also, f is defined as:

$$f(t_x, t_y) = \begin{cases} \lambda_B^2 & \text{if } t_x = t_y, \lambda_B \in (0,1) \\ 0 & \text{otherwise} \end{cases}$$

where t_x and t_y are two *terms*.

N-gram Kernel

In n-gram kernel, a string s is defined from alphabet Σ of $|\Sigma|$ symbols, and is presented in a feature space F, where each dimension is a string (Shawe-Taylor & Cristianini, 2004; Lodhi et al., 2002). Also, Σ^* represents the set of all strings and Σ^n represents the string set of length n. Furthermore, ss' represents the concatenation of strings s and s'. Now, the *substrings*: u, v_1, v_2 of string s, are defined such that:

$$s = v_1 u v_2,$$

where, if $v_1 = \varepsilon$ (ε is the empty string of 0 length) then, u is called to be the *prefix* of s and if $v_2 = \varepsilon$, then u is called to be the *suffix* of s. Now, a feature map Φ is defined in feature space F, with below embedding,

$$\Phi_u^n(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|, u \in \Sigma^n.$$

The associated kernel is defined as:

$$K_n(s,t) = \langle \Phi^n(s), \Phi^n(t) \rangle = \sum_{u \in \Sigma^n} \Phi_u^n(s) \Phi_u^n(t).$$

Also, the computational complexity of *n-gram* kernel is written as $O(n|s||t|)$ (Shawe-Taylor & Cristianini, 2004).

2.3 Parameter Optimization for Numeric Kernels

According to the redefinition of the kernel function in section 2.2.1, a novel kernel can be defined to satisfy the finitely positive semi-definite property. This allows

to have large number of feasible kernel functions, where the kernel matrix is positive semi-definite. These kernel functions have their own individual parameters. For example: polynomial kernel has parameter gamma (γ) and coefficient, gaussian kernel has parameter sigma (σ), there can be different orders of ANOVA kernels, the Von Neumann diffusion kernel is bound with parameter lambda (λ), substring length in the n-gram kernel derives different kernel matrices, and subsequence size in the fixed length subsequences kernel also derives different kernel matrices (Hofmann, Scholkopf, & Smola, 2008; Shawe-Taylor & Cristianini, 2004). As different kernel matrices yield different classification accuracies, it is vital in pattern recognition with SVM to identify the optimum parameter combination that produces the kernel matrix, which yields the highest accuracy. This optimum parameter identification can be quite challenging, if the kernel function has more than one parameter, where the optimization method has to search in a high dimensional parameter space. Adding to the complexity, the SVM cost parameter also affects the classification accuracy of a pattern recognition task. This requires one to use various optimization techniques, to identify optimum parameters for both SVM and kernel function.

There are several SVM optimization techniques discussed in the literature, namely trial and error method, grid optimization, leave-one-out cross validation, generalization error estimation using the gradient descent method, and evolutionary algorithms. In the trial and error method, a set of parameters are selected empirically, and the combination yielding the least error are taken into consideration (Imbault & Lebart, 2004). However, the method dose not produce good results, due to its impreciseness and unreliability (Zhang & He, 2010). Grid parameter search does a grid search with a fixed step-size over the parameter space, while the performance of each parameter combination is assessed via a performance measurement. This method is efficient in situations where the number of parameters are less. In practice, it requires complex computation and therefore can be time consuming (Friedrichs & Igel, 2005). The leave-one-out cross validation method has the disadvantage of being computationally expensive (Chapelle et al., 2002). The gradient descent method, which minimizes the estimates of the generalization error via gradient descent algorithm is prone to fall into a local 'minima' (Zhang & He, 2010). Furthermore, swarm optimization (Souza, Carvalho, Calvo, & Ishii, 2006), ant colony optimization (Zhang & He, 2010) and evolutionary algorithms, (particularly genetic algorithms) (Friedrichs & Igel, 2005) are also being used to optimize SVM parameters.

2.4 Parameter Optimization for String Kernels

As one can see from section 2.2.2, string kernels also have different parameters like λ_L in edit-distance kernel, λ_B in bag-of-words kernel, substring size in n-gram kernel and subsequence size in fixed length subsequence kernel, which derives different kernel matrices. Hence, the parameter optimization problem discussed in section 2.3 applies to string kernels as well. Adding to the complexity, the computation cost for strings can be quite expensive compared to numeric data, where most of the string kernels require an internal function to map strings to numbers (i.e. function r in edit-distance and function f in bag-of-words kernel that is explained in section 2.2.2). Also, if one observes the experimental results mentioned in Rieck and Laskov (2007), Lodhi et al. (2002), Sharma et al. (2007) and S.Sonnenburg (2006) even the same string kernel requires different parameter combinations on different string datasets to yield good classification accuracies. This brings about the point that string dataset characteristics also need to be included in a string kernel SVM optimization method.

Even though the methods mentioned in section 2.3, optimizing SVM parameters (mainly on numeric kernels), to our knowledge, none have focused their attention on optimize SVM string kernel parameters, in particular by taking string dataset characteristics in to consideration. Chapter 4 explains a novel method of string kernel SVM optimization, considering sting meta-features, meta learning and regression techniques.

2.5 Summary

SVMs are able to detect patterns in a high dimensional feature space without a computational decrease. Kernels are used to present data in high dimensional spaces, in situations where the pattern recognition algorithm requires only the inner product of the feature map (not the feature map itself). A kernel can be derived to have the finitely positive semi-definite property. There are many types of kernels available at the moment. Specially, string kernels are used with SVMs for string classification. Pattern recognition ability of SVMs depends upon the kernel matrix, where different parameters in the kernel function produce different kernel matrices. Also, it is effected by penalty parameter in SVM (SVM cost). Various optimization techniques

are elucidated in the literature to optimize numeric kernels with SVMs. But there is little or no literature pertaining to string kernel SVM optimization, particularly taking string dataset characteristics in to account.

Chapter 3

Motivation of Meta Learning for String Categorization

3.1 Introduction

This chapter discusses the motivation to use meta learning for string classification. The section 3.2 defines and explains meta learning, then elucidates the process of meta learning and its applications, followed by an explanation on meta-features. Section 3.3 discusses the application of meta learning for text categorization using a set of text meta-features, which are calculated from text data. Section 3.4 clearly demarcates the difference between text and string data, clarifying their ability to use some of the text meta-features for string classification in section 3.5.

3.2 Meta Learning

As Giraud-Carrier, Vilalta, and Brazdil (2004) explain, meta learning is the process of acquiring and exploiting meta-knowledge through *re-learning* from meta-features. Re-learning, which is to maintain the learning algorithm unchanged or to modify it, helps the learning system to profit from repetitive use of similar tasks. It can be applied on a single learning system to optimize parameters, or on a set of algorithms to select the best algorithm for a given classification task (Vilalta, Giraud-Carrier,

Brazdil, & Soares, 2004; Furdík et al., 2008; Brazdil, Soares, & Da Costa, 2003) (Giraud-Carrier et al., 2004). The process of re-learning requires a set of domain specific characteristics so called 'meta-features' to evaluate the performance of a algorithm or algorithms (Brazdil et al., 2003). In practice, meta learning is used to select the best algorithm for a text classification (Lam & Lai, 2001; Furdík et al., 2008), predict optimum parameters for kernels in SVMs (Soares, Brazdil, & Kuba, 2004), and to optimize neural networks (Kordk et al., 2010).

Apart from parameter optimization, meta learning on a single learning system is used to evolve the architecture of the learning system via experience, such as evolving a decision tree using past experience (Brazdil, Giraud-Carrier, Soares, & Vilalta, 2008) or to evolve a neural network considering past topology parameters (Kordk et al., 2010). On the other hand, meta learning on a set of algorithms is used in situations such as algorithm ranking (Brazdil et al., 2003) and algorithm identification in text categorization (Furdík et al., 2008). Sound meta features that effectively describe domain characteristics are required in both these systems (single learning and systems with multiple algorithms) (Brazdil et al., 2008; Giraud-Carrier et al., 2004).

Generally, there are three types of meta-features. Firstly, *simple statistical and information-theoretic meta-features* are calculated from the dataset, such as number of classes, number of features, degree of correlation between features, and average class entropy (Brazdil et al., 2008). Secondly, there are *model based meta-features*: which describe certain characteristics of the learning system, such as maximum number of nodes per feature in a decision tree, kernel width of the gaussian kernel, or maximum depth of a decision tree. Thirdly, *landmark meta-features* describe the performance (i.e., accuracy, mean squared error) of a learning algorithm (Kordk et al., 2010; Brazdil et al., 2008).

Section 3.3 of the thesis discusses previous work using meta learning for text categorization. Furthermore, section 3.5 explains the motivation for using meta learning for string classification.

3.3 Review of Meta Learning for Text Categorization

Lam and Lai (2001) explain nine text meta-features for text categorization, later expanded by Furdík et al. (2008) in their work. Both the studies by Lam and Lai (2001) and Furdík et al. (2008) use the text meta-features to build a meta model, which selects the best algorithm for a given document category in document categorization.

3.3.1 Text Meta-features

The text meta-features elucidated by Lam and Lai (2001) and Furdík et al. (2008) for document categorization are:

1. **TraningInstancesPerCategory**: Number of positive training instances per category.
2. **TestingInstancesPerCategory**: Number of positive testing instances per category.
3. **AvgDocLenPerCategory**: The average document length of a category. The document length refers to the number of index terms in a document. The average is taken across all the positive documents within a category.
4. **AvgTermValPerCategory**: The average term weight of a document within a category. The average index term weight is taken for single document and the average is then computed for all the documents in a category.
5. **AvgMaxTermValPerCategory**: The average maximum term weight of a document within a category. The maximum index term weights for individual documents are summed and the average is taken for a category.
6. **AvgMinTermValPerCategory**: The average minimum term weight of a document within a category. The minimum index term weights for individual documents are summed and average is taken for a category.

7. **AvgTermThrePerCategory**:The average number of terms above a term weight threshold for a given category. The 'term weight threshold' is set globally. The number of index terms above the term weight threshold are summed for category, and the average is computed for all instances in the category.

8. **AvgTopInfoGainPerCategory**:The average information gain of the top n index terms of a category. The information gain of each individual index term is computed for each category and ranked. The average is taken across top n index terms with the highest information gain within a category.

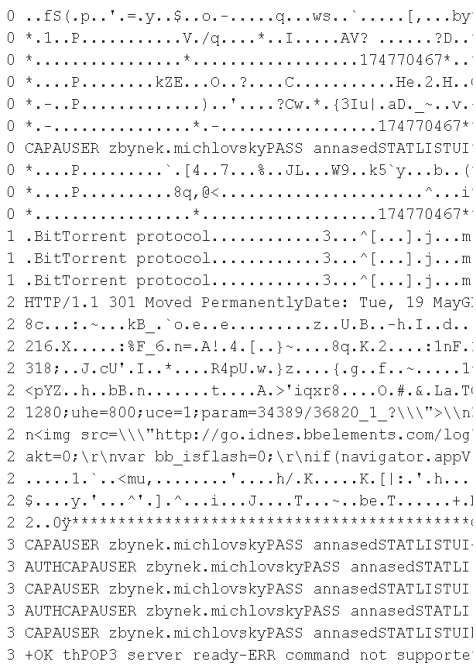
9. **NumInfoGainThresPerCategory**:The number of index terms in a category, where the information gain value exceeds a globally specified threshold.

The above text meta-features explained by Lam and Lai (2001) and Furdík et al. (2008) extracts statistical and information-theoretic information from the dataset, and later used them to train a meta model, which identifies the most suitable algorithm for a given document category.

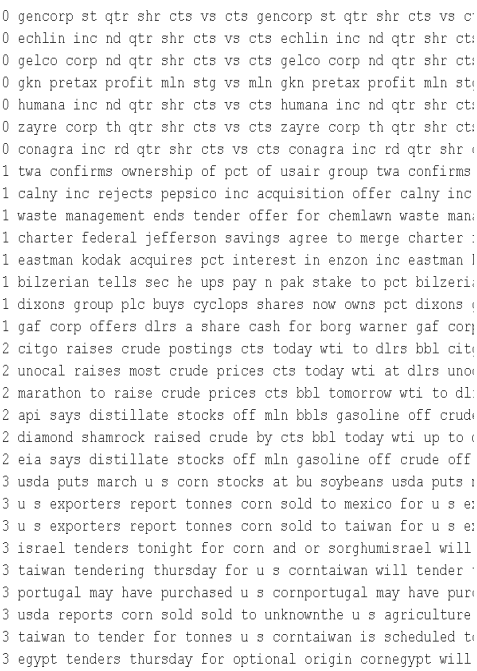
3.4 Text Data versus String Data

As the main focus of our research is to optimize string kernel SVMs, this section attempts to clearly define the difference between text and string data. According to Singhal (2001), De-Bie and Cristianini (2004) and Shawe-Taylor and Cristianini (2004) a text dataset is a collection of *words*, where word or synonymously term is any sequence of letters separated by punctuation or spaces. On the other hand, a *string* is a finite sequence of *symbols* from an *alphabet* (Shawe-Taylor & Cristianini, 2004). This means even the word demarcation symbols like space and punctuation can be in a string.

If one is given a string and a text dataset, it is easy to categorize them both into their respective data types, considering their *term* separability (De-Bie & Cristianini, 2004). This can be illustrated more clearly using Figure 3.1 where there are three types of data: network data (Figure 3.1a), Reuters-21578 news data (Figure 3.1b), and spam data which consists of spam and non-spam emails (Figure 3.1c). It is



(a) Network Data



(b) Reuters-21578 Data

(c) Spam Data

Figure 3.1: *String and Text Data: (a) Network traffic data produced by network applications. (b) Data from Reuters-21578 dataset. (c) Spam data which consists of ham and Spam E-mail messages (in every instance, the first character represents the class label followed by actual data)*

difficult to separate *terms* in both network data and spam data compared to Reuters-21578 data, but, *terms* are easily separable in Reuters-21578 data. This helps one to categorize both network data and spam as string data and reuters-21578 as text data.

By keeping this difference between string data and text data in mind, section 3.5 explains our motivation to use text meta-features explained in section 3.3.1 for string classification.

3.5 Motivation of Meta Learning for String Categorization

In section 3.3.1, except for meta features *TraningInstancesPerCategory* (1) and *TestingInstancesPerCategory* (2) all other text meta-features are computed using terms and their frequencies. In this way one is able to represent a string dataset as a collection of terms and their frequencies, which helps to derive some string meta-features that are computed by terms and their frequencies. Now these string meta-features help to employ meta learning on string classification.

3.6 Summary

Meta learning enables the exploitation of meta knowledge via *re-learning* using a set of meta-features. Lam and Lai (2001) explain a set of *text meta-features*, which enable the use of meta learning for algorithm selection in text classification. These *text meta-features* are computed using *terms* and their frequencies in a text dataset. Presenting a string dataset like *terms* and their frequencies enables it to be transformed to a new space. The *string meta-features* can be computed by the points in this new space, where it enables meta learning for string classification.

Chapter 4

Meta Learning for String Kernel SVM Optimization

4.1 Introduction

Chapter 3 explains the applicability of meta learning for model selection in text categorization. This chapter elucidates the applicability of meta learning for string kernel SVM optimization. Section 4.2, explains a mechanism to represent a string dataset in a new space, in which the points help to calculate string meta-features. Which in turn help to employ meta learning for string classification, discussed in section 4.3.1. Based upon this principle, section 4.3.2 explains the meta learning algorithm for string kernel SVM optimization.

4.2 String Meta-features

In order to use the meta-features as discussed in section 3.3.1 for string classification, the string dataset needs to be presented as terms and term frequencies. We accomplish this in a string dataset by using splitting characters: "+':(){}[]. ,-\" to split a string into set of terms or synonymously *tokens*. This approach is referred as 'tokenization' in the literature (Shawe-Taylor & Cristianini, 2004). To explain tokenization, consider the highlighted string in Figure 4.1, which is from a network

Figure 4.1: *String Data: Network traffic data produced by network applications (in every instance, the first character represents the class label of the network application, followed by actual network traffic data)*

application that uses *http* protocol. Using specified splitting characters, one can split the string into tokens: "$akt = 0$;", "r", "$nvarbb_isflash = 0$;", "r", "nif", "*navigator*", and "*appV*". Now a *token frequency table* is generated, as shown in Table 4.1. This token-frequency information is used to compute the sting meta-features explained in this section.

The main difference between the text meta features discussed in section 3.3.1 and sting meta-features explained here, is that the *text meta-features* are calculated for a text category in a text dataset, but, the *sting meta-features* are calculated for the entire string dataset. Out of nine text meta-features discussed in the section 3.3.1, seven are considered in deriving these string meta-features.

Assume a string dataset which has n number of instances, the seven sting meta-features are:

1. **AvgInstanceLen**: The average instance length of the dataset. The instance length refers to number of tokens in an instance. The average is taken across all the instances. If i^{th} instance has N_i tokens, then the average instance length

Token	Token Frequency
$akt = 0;$	1
r	2
$nvarbb_isflash = 0;$	1
nif	1
$navigator$	1
$appV$	1

Table 4.1: *Token Frequency Table: Tokens and their frequencies for highlighted string in Figure 4.1*

for that dataset is $\frac{\sum_{i=1}^{n} N_i}{n}$.

2. **AvgTokenVal**: The average token weight of an instance across a string dataset. Initially, the token weight is calculated for each token and the average is computed for single instance. Then, the average token weights for each instance are summed and the average is computed for all the instances.

   If there are m unique tokens in i^{th} instance, the average token weight for a string dataset is written as:

$$\textit{Average token weight of the string dataset} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} TW(j,i)}{mn}, \tag{4.1}$$

   where $TW(j,i)$ is the token weight of j^{th} token in i^{th} instance. According to Hersh (2008)s interpretation of *term weight*, the $TW(j,i)$ can be written as:

$$TW(j,i) = \textit{TF(j,i)} \times \textit{IDF(j)}, \tag{4.2}$$

   where *IDF(j)* is the inverse document frequency of j^{th} token, and *TF(j, i)* is the frequency of j^{th} token in instance i. Furthermore, according to Hersh (2008), the *IDF(j)* is computed as:

$$IDF(j) = \log \frac{n}{TF(j)} + 1, \tag{4.3}$$

   where $TF(j)$ is the frequency of the j^{th} token in the dataset. Now, considering

(4.2) and (4.3), equation (4.1) is rewritten as:

$$\textit{Average token weight of the string dataset} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} TF(j)\left(\log\frac{n}{TF(j)}+1\right)}{mn} \tag{4.4}$$

3. **AvgMaxTokenVal**: The average maximum token weight of an instance across a string dataset. Maximum token weights of an instance are summed and the average is taken across all instances.

4. **AvgMinTokenVal**: The average minimum token weight of an instance across a given string dataset. Minimum token weights of an instance are summed and the average is taken across all instances.

5. **AvgTokenThre**: The average number of tokens above a token weight threshold for a given string dataset. The token weight threshold is set globally. The number of tokens where the token weight is above the threshold are summed and the average is taken across all instances.

6. **AvgTopInfoGain**: The average information gain of the top r tokens in the string dataset. The information gain of each individual token is computed for each instance and raked. Then, the average is taken across top r terms with highest information gain.

7. **NumInfoGainThres**:The average number of tokens in an instance where the information gain value exceeds a globally specified threshold.

4.3 Meta Learning for String Classification

The mentioned string meta-features help to employ meta learning on a string classification. The principle of using meta learning for string classification is discussed in section 4.3.1. A novel string kernel SVM optimization method is elucidated in section 4.3.2.

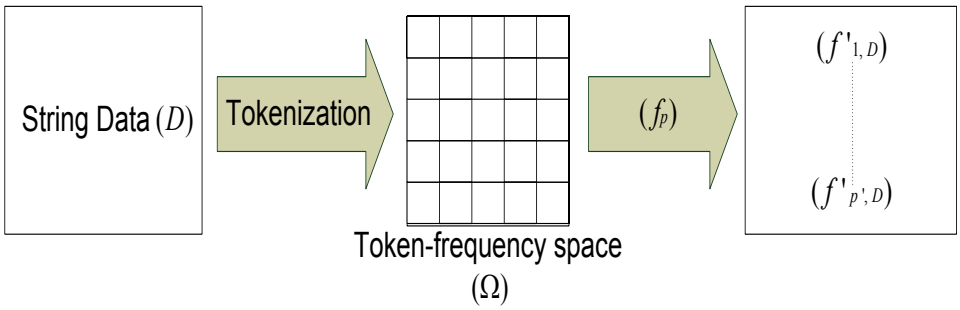


Figure 4.2: *String Meta-feature Generation Process*

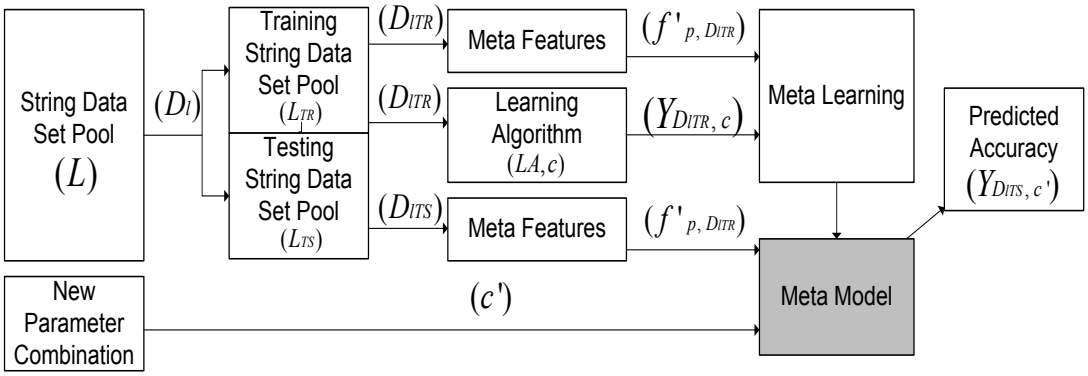


Figure 4.3: *The Procedure to Employ Meta Learning for String Classification*

4.3.1 Meta Learning for String Classification: Principle

Consider a string dataset D, which is represented as a vector in *token-frequency space* Ω, where each dimension in Ω is associated with one *token*. Now, the dataset D is represented via function ω in this new Ω token-frequency space:

$$\omega(D) = (TF(t_1, D), TF(t_2, D), ..., TF(t_N, D)) \in \Omega,$$

where $TF(t_j, D)$ is the token frequency of j^{th} token in the string dataset D, and N is the number of unique tokens in the dataset. Now one can derive a function $f_p : \Omega \rightarrow \mathbb{R}$:

$$f_p(D) = f'_{p,D}$$

where $f'_{p,D}$ represents the value for the p^{th} string-meta feature for D. For the string dataset D, there are p' finite meta-features, where all string meta-features $f_p(D)$ $(p = 1, 2, 3, ..., p')$ are well defined. This sting meta-feature generation process is shown in Figure 4.2.

Using the above discussed sting meta-features, Figure 4.3 explains the principle of meta learning for string classification. Assume there is a string dataset pool L with l' datasets, where, each string dataset D_l $(D_l \in L, l = 1, ..., l')$ is again subdivided into unique D_{lTR} (training) and D_{lTS} (testing) datasets, which creates training (L_{TR}) and testing (L_{TS}) dataset pools. The string meta-feature $f'_{p,D_{lTR}}$ is computed for dataset D_{lTR}. Also, for D_{lTR}, the machine learning algorithm LA with parameter combination c, generates $Y_{D_{lTR},c}$ classification accuracy. These computed string meta-features ($f'_{p,D_{lTR}}$), parameter combinations (c) and accuracy information ($Y_{D_{lTR},c}$) generate a meta model via regression, which is able to predict the classification accuracy for a new string dataset, given the computed sting meta-features and the parameter combination. Hence, for a new string dataset D_{lTS}, the meta model predicts the accuracy $Y_{D_{lTS},c'}$ for parameter combination c' by computing string-meta features $f'_{p,D_{lTS}}$.

4.3.2 Meta Learning for String Kernel SVM Optimization: Algorithm

According to the principle introduced in section 4.3.1, the built meta model is able to predict the string classification accuracy for a machine learning algorithm on a novel string dataset, using computed string meta-features. This section explains the procedure to use this principle (meta learning for string classification) to optimize string kernel SVMs, which is shown in Algorithm 1.

Algorithm 1 explains the procedure to use meta learning to optimize string kernel SK with SVM. The proposed algorithm uses training string dataset pool L_{TR}, testing string dataset pool L_{TS}, training parameter pool C and testing parameter pool C' as inputs. Also, SVM with string kernel SK is set as the learning algorithm (LA). Initially, a meta model is built using meta features calculated for each dataset in L_{TR} with accuracy information obtained for each parameter combination in C via n-fold cross validation. Then, for each new string dataset in L_{TS}, the built meta

input :
L_{TR} =Training String Dataset Pool
L_{TS} =Testing String Dataset Pool
C =Parameter Combination Pool for Training ($c \in C$)
C' =Parameter Combination Pool for Testing ($c' \in C'$)
LA =SVM with String Kernel SK
output: Parameter combination $\hat{c}_l$ which yields the best accuracy for sting dataset D_{lTS}

for $l \leftarrow 1$ **to** l' **do**
  Pick D_{lTR} from L_{TR}
  for $p \leftarrow 1$ **to** p' **do**
    Compute $f'_{p,D_{lTR}}$
  end
  repeat
    Pick a parameter combination c from C
    Do 10-fold cross validation on D_{lTR}, using LA with parameter combination c which yields $Y_{D_{lTR},c}$ accuracy
  until *no more parameter combinations in* C;
end

Build a regression model (*meta model*) using $f'_{p,D_{lTR}}$, c, and $Y_{D_{lTR},c}$
for $l \leftarrow 1$ **to** l' **do**
  Pick D_{lTS} from L_{TS}
  for $p \leftarrow 1$ **to** p' **do**
    Compute $f'_{p,D_{lTS}}$
  end
  repeat
    Pick a parameter combination c' from C'
    Predict accuracy $Y_{D_{lTS},c'}$ for LA with parameter combination c' using build *meta model*
    if $Y_{D_{lTS},c'}$ *is maximum* **then**
      $\hat{c}_l = c'$
    end
  until *no more parameter combinations in* C;
end

Algorithm 1: The Proposed Meta Learning Algorithm for String Kernel SVM Optimization

model predicts the classification accuracy for each combination in C'. The combination ($\hat{c}_l$) which yields the highest accuracy is presented as the optimum parameter combination for dataset D_{lTS}.

4.4 Summary

A string dataset is defined in a new space using *tokens* and their frequencies (*token-frequency space*). A set of *string meta-features* is computed by the points of this new space. These string meta-features, learning algorithm parameters and accuracy information help to train a *meta model* which predicts string classification accuracy for a learning algorithm on a new string dataset. This principle is used to optimize string kernel SVMs in the proposed algorithm (Meta Learning Algorithm for String Kernel SVM Optimization).

Chapter 5

Experiments and Results

The chapter presents the results of experiments carried out to demonstrate the effectiveness of the proposed meta learning algorithm for string kernel SVM optimization. Three string kernel SVMs: *edit-distance* SVM, *bag-of-words* SVM and *n-gram* SVM were experimented, on four string datasets: Spam, Reuters-21578, Network Application Detection and e-News categorization using the proposed algorithm. Section 5.1 explains the experimental setup, evaluation criteria and dataset description. The experiment results for the three string kernel SVMs are presented in section 5.2.

5.1 Experiments

The experiment setup, the evaluation criteria and string datasets are explained in this section.

5.1.1 Experiment Setup

The proposed algorithm was experimented on three string kernel SVMs (*edit-distance* SVM, *bag-of-words* SVM and *n-gram* SVM). As shown in Table 5.1, the algorithm was trained using training string dataset pool L_{TR}, and was tested on testing string dataset pool L_{TS}, for each string kernel SVM. In the experiments, SVM cost parameter (c) was selected as $2^0, 2^1, ..., 2^{16}$ for all string kernel SVMs. The string kernel

parameters λ_L and λ_B for edit-distance and bag-of-words string kernels, were selected as $0.001, 0.1, 0.25, 0.5$. The substring length in n-gram string kernel was selected as $1, 2, ..., 8$ in the experiments. The string meta-feature, $AvgMinTokenVal$ was not considered in the training stage, as it was having the value 0 for all datasets. Also, the global threshold for the $AvgTokenThr$ was set to 2 in all the experiments. Support Vector Regression (SVR) was used to build the meta model. In the training stage, the parameters which yield lowest cross validation RMSE for SVR, were considered in regression (in building the meta model), for each string kernel SVM. 10 fold- cross validation was done for the top 10 predicted parameter combinations, on each string dataset. The performance evaluation was done considering Root Mean Squared Error (RMSE) for the top 10 predicted parameters on each dataset.

In the experiments, edit-distance and bag-of-words string kernels were implemented using Libsvm-2.9 Chang and Lin (2001) and n-gram string kernel was implemented using shogun octave interface (S.Sonnenburg, 2006). The string meta-feature computation program was coded using C++ language. All the experiments were run on a PC having Intel Core2 Duo 3GHz processor and 2.96 Gb RAM.

5.1.2 Datasets

Four string datasets were used in the string dataset pool $L = \{$*Spam*, *Reuters-21578*, *Network Application Detection*, *e-News Categorization*$\}$. It was again subdivided into training dataset pool (L_{TR}) and testing dataset pool (L_{TS}), where each consisted of unique string datasets. String dataset pool L_{TR} was used to train the meta model in the proposed algorithm and string datasets pool L_{TS} was used to test the proposed algorithm. For edit-distance and bag-of-words string kernel SVMs, training dataset pool $L_{TR} = \{$*Spam*, *Reuters-21578*, *Network Application Detection*$\}$ was used, and for n-gram SVM, $L_{TR} = \{$ *Spam*, *Reuters-21578*, *Network Application Detection*, *e-News Categorization* $\}$ was used. In testing, for edit-distance and bag-of-words string kernel SVMs, testing pool $L_{TS} = \{$*Spam*, *Reuters-21578*, *Network Application Detection*$\}$ was used, and for n-gram SVM, the testing pool $L_{TR} = \{$ *Spam*, *Reuters-21578*, *Network Application Detection*, *e-News Categorization* $\}$ was used. Table 5.1 summarizes algorithm training and testing information on each string dataset. A detailed description about each string dataset is given below.

Dataset Pool	String Dataset (Dataset Label)	Sting Kernel SVM		
		Edit-Distance	Bag-of-Words	N-gram
L_{TR}	Spam	✓	✓	✓
	Reuters-21578	✓	✓	✓
	Network Application Detection	✓	✓	✓
	e-News Categorization	-	-	✓
L_{TS}	Spam(1)	✓	✓	✓
	Reuters-21578(2)	✓	✓	✓
	Network Application Detection(3)	✓	✓	✓
	e-News Categorization(4)	-	-	✓

Table 5.1: *The String Datasets used in Training and Testing the Proposed Algorithm*

Class Label	Document Category	Training	Testing
0	Earn	152	40
1	Acquisition	114	25
2	Crude	76	15
3	Corn	38	10
		370	90

Table 5.2: *Data Distribution-Reuters-21578 Dataset*

Spam Dataset

This dataset consists of 696 ham messages and 384 spam messages from *Spam Assassin public mail corpus* (2002). There are two types of ham e-mails: easy ham (646) and hard ham (50). Easy ham e-mails are non-spam messages without any spam signatures and hard ham are non-spam messages similar in many aspects to spam messages which use unusual HTML markup, coloured text, spam-sounding phrases, etc. Each e-mail message has a header, a body and some potential attachments. The training dataset consists with 810 messages (484 easy ham, 38 hard ham and 288 spam) and testing dataset has 270 messages (162 easy ham, 12 hard ham and 96 spam).

Class Label	Application/protocol	Training	Testing
0	AIM	18	7
1	Bittorrent	140	59
2	http	583	249
3	pop	17	7
		770	329

Table 5.3: *Data Distribution-Application Detection Dataset*

Reuters-21578 Dataset

The Reuters dataset used in the experiments has the exact split to Lodhi et al. (2002). It consists of 470 documents: 380 for training and 90 for testing. Four document categories, those of earn, acquisition, crude and corn are available in the dataset. Table 5.2 shows the document distribution among the different categories.

Network Application Detection Data

The dataset consists of network traffic data produced by network applications, such as http, https, imap, pop3, ssh, ftp and bittorent. All network data were captured, and sorted according to their protocols using "Wireshark" (Combs et al., 2008) and split into individual connections using *tcpflow* (Elson, 2003). Only TCP traffic was taken into account in the data capturing stage. The option '-s' in "tcpflow" (Elson, 2003) was used to remove all non printable characters in a connection. Also, only the first 50 bytes of a connection were considered in preparing the dataset. Every connection was labelled according to the application type. Table 5.3 shows the label number for every application type and the number of instances in training and testing datasets.

e-News Categorization Data

The dataset is collected from four electronic newspapers: *New Zealand Herald* (2010), *The Australian* (2010), *The Independent* (2010) and *The Times* (2010), on five news topics (business, education, entertainments, sport and travel). Each document is labelled manually by skimming over the text to identify the category. Punctuations

Class Lable	News Category	Training	Testing
0	Business	227	97
1	Education	93	40
2	Entertainments	99	42
3	Sport	118	50
4	Travel	131	56
		668	285

Table 5.4: *Data Distribution e-Newsgroup Dataset*

and stop words were removed from the dataset in advance. Table 5.4 shows detailed information about the dataset.

5.2 Results

Experiment results for three string kernel SVMs are discussed in this section. Section 5.2.1 explains the results for *edit-distance* SVM optimization, using the proposed algorithm. The experimental results for *bag-of-words* SVM optimization are presented in section 5.2.2. Section 5.2.3 explains the results for *n-gram* SVM optimization, using the proposed algorithm.

5.2.1 Meta Learning for *Edit-Distance* SVM Optimization

Here, the proposed algorithm was used to optimize edit-distance SVM. The algorithm attempts to find the optimum parameter combination (λ_L and SVM cost) for *edit-distance* SVM on three string datasets: spam, Reuters-21578 and network application detection, in test dataset pool L_{TS} (refer Table 5.1). In the experiments, the SVR parameters: $\gamma = 0.084$ and *SVR Cost=5400* were used in regression. The actual accuracy and the predicted accuracy for the top 10 predicted parameter combinations are shown in Table 5.5. Also, the table shows the RMSE for top 10 predicted parameter combinations on each dataset.

According to Table 5.5a and Table 5.5b, the optimum parameters produced by the proposed algorithm yield very low predicted and actual classification accuracies, for *edit-distance* SVM, on spam and Reuters-21578 string datasets. This shows that

cost	λ_L	rank	predicted%	actual%
65536	0.000488	1	2.36387	0.37
2	0.000488	2	2.36381	5.19
4	0.000488	3	2.36377	4.44
8	0.000488	4	2.36371	3.33
16	0.000488	5	2.36357	1.11
32	0.000488	6	2.36331	0.37
2	0.000976	7	2.36302	4.44
4	0.000976	8	2.36299	3.33
8	0.000976	8	2.36292	1.11
65536	0.000976	10	2.36286	0.37
root mean squared error				1.831455

(a) Spam Data

cost	λ_L	rank	predicted%	actual%
65536	0.000488	1	3.30463	23.33
2	0.000488	2	3.30454	24.44
4	0.000488	3	3.30451	26.67
8	0.000488	4	3.30445	23.33
16	0.000488	5	3.30432	23.33
32	0.000488	6	3.30407	23.33
2	0.000976	7	3.3038	26.67
65536	0.000976	8	3.30379	23.33
4	0.000976	9	3.30377	23.33
8	0.000976	10	3.30371	23.33
root mean squared error				20.846794

(b) Reuters-21578

cost	λ_L	rank	predicted%	actual%
32768	0.0625	1	73.1729	75.99
32768	0.03125	2	73.1653	75.99
32768	0.125	3	73.1602	75.99
32768	0.015625	4	73.1556	75.99
8192	0.125	5	73.1526	75.99
16384	0.125	6	73.1526	75.99
8192	0.25	7	73.1518	75.99
32768	0.5	8	73.1515	75.99
4096	0.25	9	73.1514	75.99
65536	0.25	10	73.1512	75.99
root mean squared error				2.833499

(c) Network Application Detection

Table 5.5: *Experimental Results for Edit-Distance SVM Optimization: (the top 10 predicted parameter combinations using the proposed algorithm on each string dataset)*

the *edit-distance* SVM is not suitable for string classification on spam and reuters-21578 datasets. However, according to Table 5.5c, the algorithm produces optimum parameters which yield good string classification accuracies on network application detection dataset, for *edit-distance* SVM (with a low RMSE).

5.2.2 Meta Learning for *Bag-of-Words* SVM Optimization

Here, the the proposed algorithm attempts to find the optimum parameter combination (λ_B and SVM cost) for *bag-of-words* SVM. Initially, the algorithm was trained on a training dataset pool L_{TR} (refer Table 5.1). Then, the algorithm predicted the string classification accuracies for *bag-of-words* SVM on three different string datasets in test dataset pool L_{TS} (refer Table 5.1). The SVR parameters: $\gamma = 0.88$ and *SVR Cost=450* were used in regression. 10-fold cross validation was done for the top 10 predicted parameter combinations, and RMSE was calculated for the same. Table 5.6 shows the top 10 predicted parameter combinations on three string datasets for *bag-of-words* SVM.

cost	λ_B	rank	predicted%	actual%
2	0.000488	1	61.6244	64.44
4	0.000488	2	61.6207	64.44
8	0.000488	3	61.6135	64.44
16	0.000488	4	61.5989	64.44
32	0.000488	5	61.5699	64.44
64	0.000488	6	61.5117	64.44
2	0.000976	7	61.4788	64.44
4	0.000976	8	61.4752	64.44
8	0.000976	9	61.4679	64.44
16	0.000976	10	61.4534	64.44
root mean squared error				2.899333

(a) Spam Data

cost	λ_B	rank	predicted%	actual%
4096	0.5	1	87.9839	86.67
2048	0.5	2	87.9734	86.67
1024	0.5	3	87.9551	86.67
512	0.5	4	87.9422	86.67
256	0.5	5	87.9347	86.67
8192	0.5	6	87.9307	86.67
128	0.5	7	87.9307	86.67
32768	0.5	8	87.9304	86.67
64	0.5	9	87.9287	86.67
32	0.5	10	87.9276	86.67
root mean squared error				1.273886

(b) Reuters-21578

cost	λ_B	rank	predicted%	actual%
4	0.000488	1	75.3987	75.99
2	0.000488	2	75.3987	75.99
8	0.000488	3	75.3986	75.99
16	0.000488	4	75.3985	75.99
32	0.000488	5	75.3983	75.99
64	0.000488	6	75.3978	75.99
128	0.000488	7	75.3968	75.99
4	0.000976	8	75.3967	75.99
2	0.000976	9	75.3967	75.99
8	0.000976	10	75.3966	75.99
root mean squared error				0.592261

(c) Network Application Detection

Table 5.6: *Experimental Results for Bag-of-Words SVM Optimization: (the top 10 predicted parameter combinations using the proposed algorithm on each string dataset)*

According to Table 5.6b and Table 5.6c, the proposed algorithm produces parameter combinations which yield high classification accuracies on reuters-21578 and network application detection datasets with a very low RMSE. However, on spam dataset, the optimized parameter combinations produced by the proposed algorithm yield average string classification accuracies (see Table 5.6a). Considering the overall high string classification accuracy (with a very low average RMSE) shown in Table 5.6 and Table 5.8, on all three datasets, one can say that the proposed algorithm produces optimized parameter combinations which yield good string classification accuracies, for *bag-of-words* SVM.

5.2.3 Meta Learning for *N-gram* SVM Optimization

In this experiment, the algorithm attempts to find optimized parameters (substring length and SVM cost) for *n-gram* SVM. The algorithm was trained on string dataset pool L_{TR} and tested on testing string dataset pool L_{TS} (refer Table 5.1). The SVR parameters: $\gamma = 0.95$ and *SVR Cost=500* were used in regression. 10-fold cross validation was done for the top 10 predicted parameter combinations on each

cost	substring length	rank	predicted%	actual%
16384	8	1	99.3074	98.33333
4096	8	2	99.2982	98.33333
32768	7	3	99.2683	98.33333
16384	7	4	99.2652	98.33333
4096	7	5	99.2628	98.33333
4096	6	6	99.2524	98.33333
4096	5	7	99.2066	98.33333
2048	8	8	99.1935	98.33333
32768	6	9	99.1862	98.33333
4096	2	10	99.1803	97.81251
root mean squared error				0.971167898

(a) Spam Data

cost	substring length	rank	predicted%	actual%
4096	6	1	92.6961	95.36587
2048	6	2	92.6842	95.36587
8192	6	3	92.6771	95.36587
1024	6	4	92.6636	95.36587
512	6	5	92.6474	95.36587
16384	6	6	92.6405	95.36587
256	6	7	92.6376	95.36587
128	6	8	92.6322	95.36587
64	6	9	92.6294	95.36587
32	6	10	92.6279	95.36587
root mean squared error				2.712372587

(b) Reuters-21578

cost	substring length	rank	predicted%	actual%
16384	2	1	99.6656	98.22917
2048	2	2	99.5895	98.22917
1024	2	3	99.5776	98.22917
32768	2	4	99.5685	98.22917
4096	2	5	99.5634	98.22917
512	2	6	99.5607	98.22917
256	2	7	99.5489	98.22917
128	2	8	99.542	98.22917
64	2	9	99.5384	98.22917
32	2	10	99.5365	97.81252
root mean squared error				1.386738588

(c) Network Application Detection

cost	substring length	rank	predicted%	actual%
4096	5	1	90.8189	74.35295
4096	6	2	90.6225	75.05881
4096	4	3	90.5267	73.76471
8192	5	4	90.484	73.88235
32768	5	5	90.3336	73.64706
16384	5	6	90.3317	73.64706
8192	6	7	90.3088	75.17647
4096	3	8	90.2765	74.70588
8192	4	9	90.275	73.41176
32768	4	10	90.1778	73.17646
root mean squared error				16.34502652

(d) e-News Categorization

Table 5.7: *Experimental Results for N-gram SVM Optimization: (the top 10 predicted parameter combinations using the proposed algorithm on each string dataset)*

string dataset. Table 5.7 summarizes the experiment results for the top 10 predicted combinations on each dataset.

According to Table 5.7, the proposed algorithm produces optimized parameters, which yield good string classification accuracies for *n-gram* SVM, on all four string datasets. The algorithm has a very low RMSE for top 10 predicted on spam, Reuters-21578 and network application detection datasets (see Table 5.7a, Table 5.7b, and Table 5.7c). Even though, the algorithm has quite a high RMSE on the e-News categorization dataset, the top 10 predicted parameter combinations yield good string classification accuracies on the dataset (see Table 5.7d).

5.2.4 Affection of SVR Parameters to *N-gram* SVM Optimization

This experiment attempts to identify the effect of SVR parameters on building the meta model in the proposed algorithm. The *n-gram* SVM was optimized using the proposed algorithm. *SVR Cost* was considered as 400, 450, 500, 550, 600, 650 in the experiments and *gamma* (γ) was chosen as 0.05, 0.1, 0.15,...,0.95 for SVR. All four

String Kernel	Dataset	RMSE	Avg RMSE
Edit-Distance	Spam	1.831455	8.503916
	Reuters-21578	20.846794	
	Network Application Detection	2.833499	
Bag-of-Words	Spam	2.899333	1.588493
	Reuters-21578	1.273886	
	Network Application Detection	0.592261	
N-gram	Spam	0.971168	5.353826
	Reuters-21578	2.712373	
	Network Application Detection	1.386739	
	e-News Categorization	16.345027	

Table 5.8: *Root Mean Squared Error (RMSE) for String Kernel SVM Optimization on each String Dataset (for top 10 predicted parameter combinations)*

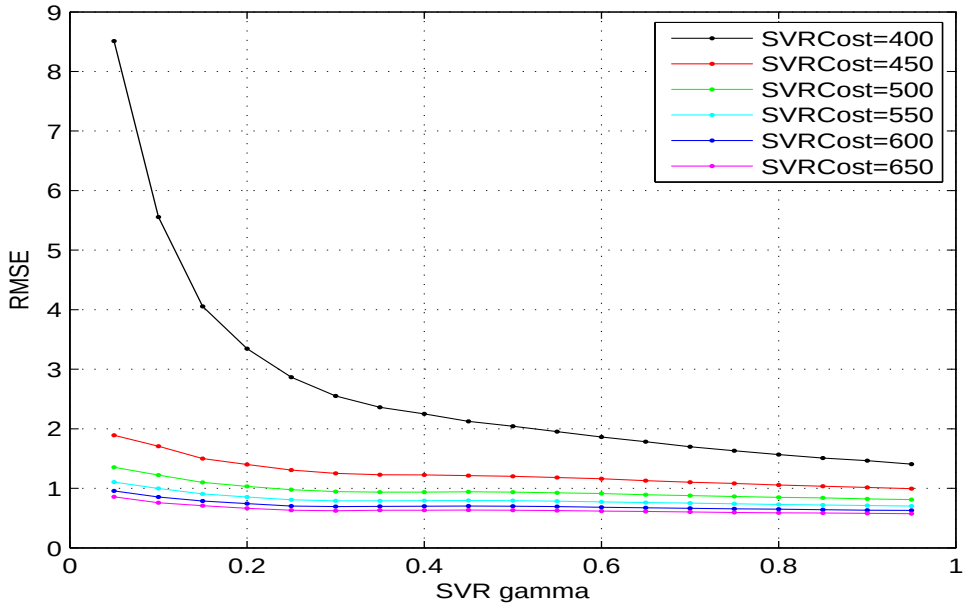


Figure 5.1: *Effectiveness of SVR Parameters in the Proposed Algorithm for N-gram SVM Optimization*

string datasets were considered in the experiment. The 10-fold cross validation was done for all 512 (4 datasets × 16 SVM cost values × 8 substring lengths) *n-gram* SVM parameter combinations on four datasets. RMSE was calculated considering actual string classification accuracy and predicted string classification accuracy for all 512 *n-gram* SVM parameters, for a given SVR parameter combination. Figure

5.1 shows the RMSEs for different SVR parameter combinations. According to the figure one can see that higher SVR cost values with higher γ values yield low RMSEs, but, the degree of affect on RMSE gets lower as SVR cost and γ increase. Detailed information regarding the relevant RMSEs can be found in Table A.1 in the Appendix A.

5.2.5 Prediction Behaviour of the Meta Learning Algorithm for *N-gram* SVM Optimization

This experiment attempts to identify the prediction behaviour of the proposed

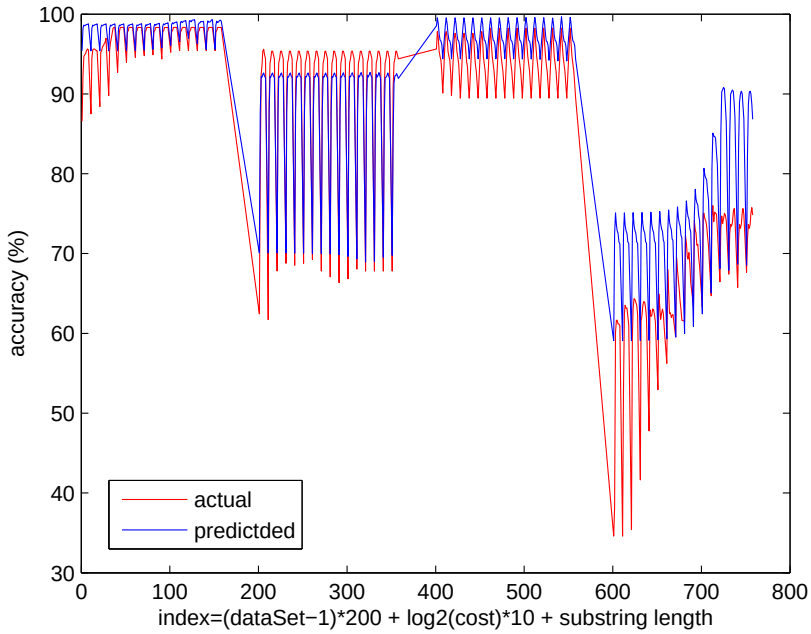


Figure 5.2: *N-gram SVM Optimization using the Proposed Algorithm for 512 Parameter Combinations (SVR-Cost=500, gamma (γ)=0.95 and RMSE=0.810938), index: $0 \leq$ Spam $< 200 \leq$ Reuters-21578 $< 400 \leq$ Network Application Detection $< 600 \leq$ e-News Categorization < 800*

algorithm when optimizing the *n-gram* SVM. Here, the 10-fold cross validation was done for all 512 (4 datasets $\times$ 16 SVM cost values $\times$ 8 substring lengths) *n-gram* SVM parameter combinations on four datasets. RMSE was calculated considering actual string classification accuracy and predicted string classification accuracy for all 512

n-gram SVM parameter combinations. SVR cost=500 and γ=0.95 were considered in the experiments.

Figure 5.2 shows actual classification accuracy and predicted string classification accuracy by the proposed algorithm, on four string datasets. In Figure 5.2, the x axis or the index is chosen in such a way that it is able to represent all parameter combinations on each dataset. This is achieved by labeling the testing datasets in dataset pool L_{TS}, according to the schema shown in Table 5.1, and then computing the index as:

$$index = (dataset - 1) \times 200 + log_2(cost) \times 10 + substring\ length.$$

Now, the index values (0,168) represent spam dataset, (200,368) represent Reuters-21578 dataset, (400,568) represent network application detection dataset and (600,768) represent e-News categorization dataset. According to the figure, one can see that the proposed algorithm nicely represents the actual parameter behaviour of *n-gram* SVM on all four datasets. This is further analyzed in Figure 5.3 considering network application dataset. According to Figure 5.3a, it can be clearly seen that substring length 2 yields good string classification accuracies on network application detection dataset. This behaviour is clearly represented by the proposed algorithm in Figure 5.3b and in Table 5.7c by producing parameter combinations that include substring length 2 in the top 10 predicted parameter combinations. (Figures that show the parameter behaviour on other string datasets can be found in the Appendix B)

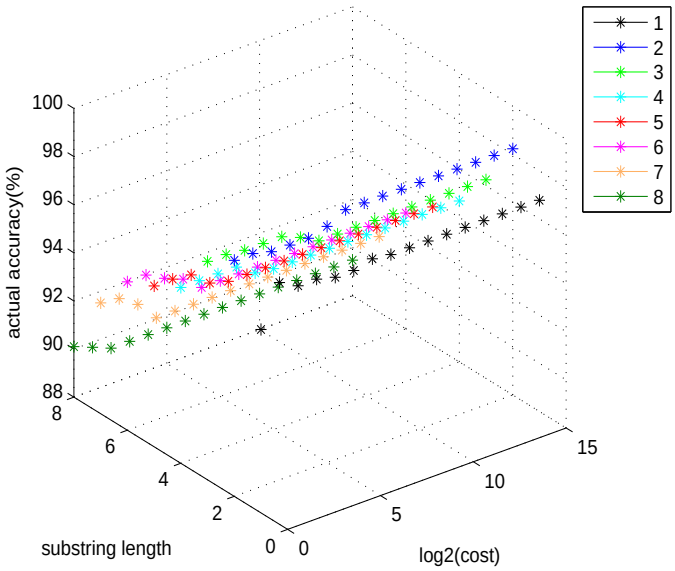
5.3 Summary

The proposed meta learning algorithm for string kernel SVM optimization produces optimized parameters for *n-gram* SVM, yielding good string classification accuracies on all four datasets. Furthermore, the algorithm produces optimized parameters which yield good string classification accuracies for *bag-of-words* SVM, on three out of two string datasets in the test dataset pool L_{TS}. The algorithm produces optimized parameters which yield good string classification accuracies only on network application detection dataset for *edit-distance* SVM.

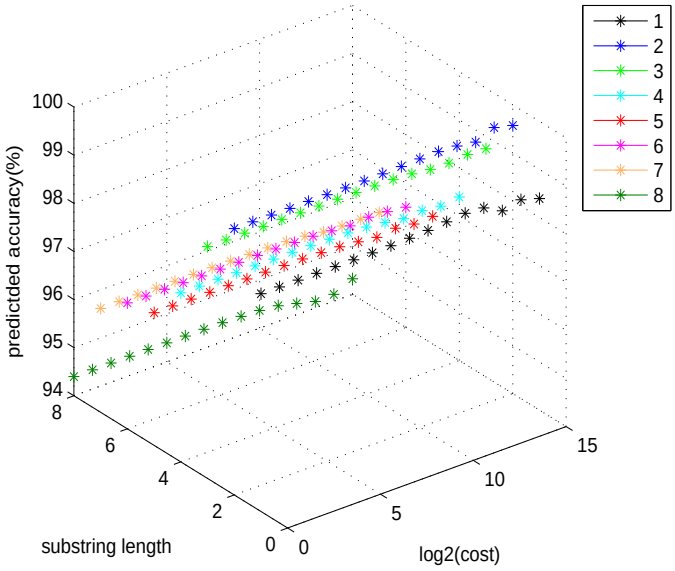
The low string classification accuracies (actual and predicted) for the top 10 param-

eter combinations, by the proposed algorithm, on spam and Reuters-21578 datasets, for the *edit-distance* SVM, reveals that *edit-distance* SVM may not be suitable for string classification on those two string datasets.

Regression parameters also have an effect on the prediction behaviour of the proposed algorithm. According to experimental results regression parameters: SVR cost=600 and γ=0.95 yields good *n-gram* SVM parameters for the proposed algorithm.



(a) Actual Parameter Behaviour: Network Application Detection (different colours represent different substring lengths)



(b) Predicted Parameter Behaviour: Network Application Detection (different colours represent different substring lengths)

Figure 5.3: *Prediction Behaviour of the Proposed Algorithm for N-gram SVM Optimization on Network Application Detection String Dataset (using SVR cost=500 and γ=0.95 in regression): (a) Actual parameter behaviour of n-gram SVM on Network Application Detection string dataset. (b) Predicted parameter behaviour of n-gram SVM, by the proposed algorithm on Network Application Detection string dataset.*

Chapter 6

Discussions and Conclusions

6.1 Brief Review of the Work

Section 1.2.1 explains the two research objectives of the research. The first is to assess the feasibility of representing a string dataset in a space similar to term-frequency space, as discussed in Lam and Lai (2001) and Furdík et al. (2008). Secondly, if a string dataset is presented in such a space, to assess whether string kernel SVMs can be optimized via meta learning?

In order to understand the principle concepts of string kernel SVMs, chapter 2 does a literature review on SVMs, kernel trick, and string kernels. The chapter also explains the SVM parameter optimization problem, along with currently available SVM parameter optimization techniques for numeric kernels. Finally the chapter discusses the string kernel SVM optimization problem.

As meta learning techniques are to be used in string kernel SVM optimization, chapter 3 discusses the principle of meta learning and its functionality. It also explains the application of meta learning for text classification using a set of text meta-features, which are calculated from text data. Furthermore, the chapter clearly defines the difference between text and string data, which motivates one to use meta learning for string classification, if there exist any defined string meta-features over string data.

The chapter 4 discusses a set of string meta-features which are defined over the token-frequency space. These string meta-features help to employ meta learning on

string classification. Based on this principle, a novel meta learning algorithm for string kernel SVM optimization is presented later in the chapter.

Chapter 5 explains the experimental results for parameter optimization of three string kernel SVMs (*edit-distance* SVM, *bag-of-words* SVM and *n-gram* SVM) using the proposed algorithm, on four string datasets.

6.2 Conclusion of the Work

The experimental results show that the proposed algorithm produces parameter combinations which yield good string classification accuracies on most of the datasets. They also reveal that some string kernel SVMs may not be suitable for carrying out the string classification required on certain string datasets. Specifically, *edit-distance* SVM yields poor string classification results on both spam and Reuters-21578 string datasets.

There are three main contributions from the thesis to the field of machine learning:

1. **String Meta-features**: The defined *string meta-features* can be used to extracting meta knowledge from any string dataset.

2. **Meta Learning for String Classification Principle**: explains the procedure to apply meta learning on string classification, using extracted meta knowledge via *string meta-features*.

3. **Meta Learning Algorithm for String Kernel SVM Optimization**: using the *Meta Learning for String Classification Principle*, a novel string kernel optimization method is derived, which is able to predict optimum string kernel SVM parameters for a given string kernel SVM on a string dataset by calculating relevant *string meta-features*.

6.3 Limitations of the Research

Below are limitations that could be found in the research.

Firstly, According to Figure A.1 in the appendix A, for *n-gram* SVM, the proposed algorithm yields classification accuracies over 100% on spam and Reuters-21578 string datasets. This can be resolved if one is able to set upper and lower bounds in the algorithm. Secondly, the *string meta-features* could be more descriptive, since, they can explain the string dataset more clearly than just numbers representing the string dataset. Thirdly, in general, a meta learning algorithm is tested on a large number of datasets, but due to the high computational cost of string data and the unavailability of bench marked string datasets, have forced to use only four string datasets in the experiments.

6.4 Future work

Even though the proposed meta learning algorithm for string kernel SVM optimization was able produce parameter combinations which yield good string classification accuracies for string kernel SVMs on most of the string datasets, one can see that some string kernel SVMs (specially *edit-distance* SVM) do not produce good classification results on certain string datasets. This can be further analyzed by doing more experiments for *edit-distance* SVM on relevant string datasets (spam and Reuters-21578). Also, instead of using string classification accuracy as the *landmark meta-feature*, one can use a *response variable*:

$$y_{ij} = ln\frac{e_{ij}}{1 - e_{ij}}$$

where, e_{ij} is the classification error for i^{th} string kernel SVM parameter combination on j^{th} string dataset (Lam & Lai, 2001). Here, the classification error (e_{ij}) is sent through a logistic transformation to yield the response variable (y_{ij}) which has the range (0,1). This avoids situations like predicting over 100% classification accuracy on a given string dataset by the proposed algorithm. Furthermore, one could implement the string meta-features: *AvgTopInfoGain* and *NumInfoGainThres* (explained in section 4.2) in a future string kernel SVM optimization algorithm. Also, identifying more string meta-features helps to employ meta learning more effectively on string classification. Finally, one could include more string datasets to the string dataset pool, where it could help to check the robustness of the proposed algorithm.

References

Aizerman, A., Braverman, E. M., & Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, *25*, 821–837.

The Australian. (2010). `http://www.theaustralian.com.au`. (Last retrieved April 14, 2010)

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–152).

Brazdil, P., Giraud-Carrier, C., Soares, C., & Vilalta, R. (2008). *Metalearning: Applications to data mining*. New York,NY: Springer-Verlag.

Brazdil, P., Soares, C., & Da Costa, J. (2003). Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, *50*(3), 251–277.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: a library for support vector machines [Computer software manual]. (Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`)

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*(1), 131–159.

Chow, R., Zhong, W., Blackmon, M., Stolz, R., & Dowell, M. (2008). An efficient SVM-GA feature selection model for large healthcare databases. In *Proceedings of the 10th annual conference on genetic and evolutionary computation* (pp. 1373–1380).

Combs, G., et al. (2008). Wireshark-network protocol analyzer. *Current July*. Available from `http://www.wireshark.org/`

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

De-Bie, T., & Cristianini, N. (2004). Kernel methods for exploratory pattern analysis: A demonstration on text data. *Structural, Syntactic, and Statistical Pattern Recognition*, 16–29.

Eitrich, T., & Lang, B. (2006). Efficient optimization of support vector machine

learning parameters for unbalanced datasets. *Journal of Computational and Applied Mathematics*, *196*(2), 425–436.

Elson, J. (2003). *tcpflow-a tcp flow recorder.* `http://www.circlemud.org/~jelson/software/tcpflow/`. (Retrieved November 29, 2009)

Friedrichs, F., & Igel, C. (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, *64*, 107–117.

Frohlich, H., & Zell, A. (2005). Efficient parameter selection for support vector machines in classification and regression via model-based global optimization. In *2005 IEEE International Joint Conference on Neural Networks, 2005. IJCNN'05. Proceedings* (Vol. 3).

Furdík, K., Paralič, J., & Tutoky, G. (2008). Meta-learning method for automatic selection of algorithms for text classification. In *Central European Conference on Information and Intelligent Systems 19 th International Conference 2008 Proceedings.*

Giraud-Carrier, C., Vilalta, R., & Brazdil, P. (2004). Introduction to the special issue on meta-learning. *Machine Learning*, *54*(3), 187–193.

Hersh, W. (2008). *Information retrieval: A health and biomedical perspective.* New York,NY: Springer Verlag.

Hofmann, T., Scholkopf, B., & Smola, A. (2008). Kernel methods in machine learning. *Annals of Statistics*, *36*(3), 1171–1220.

Imbault, F., & Lebart, K. (2004). A stochastic optimization approach for parameter tuning of support vector machines. In *Proceedings of the Pattern Recognition, 17th International conference on (ICPR'04) volume 4-volume 04* (p. 600).

The Independent. (2010). `http://www.independent.co.uk`. (Last retrieved April 11, 2010)

Kordk, P., Koutnk, J., Drchal, J., Kovrk, O., Cepek, M., & Snorek, M. (2010). Meta-learning approach to neural network optimization. *Neural Networks*, *23*(4), 568 - 582. Available from `http://www.sciencedirect.com/science/article/B6T08-4YF5R4P-1/2/fd7afbdcec93d15da3a34a5942dca697` (The 18th International Conference on Artificial Neural Networks, ICANN 2008)

Lam, W., & Lai, K. (2001). A meta-learning approach for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR conference on Research and Development in Information Retrieval* (p. 309).

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002).

Text classification using string kernels. *The Journal of Machine Learning Research*, *2*, 444.

New Zealand Herald. (2010). `http://www.nzherald.co.nz`. (Last retrieved April 12, 2010)

Rieck, K., & Laskov, P. (2007). Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, *2*(4), 243–256.

Sharma, O., Girolami, M., & Sventek, J. (2007). Detecting worm variants using machine learning. In *Proceedings of the 2007 acm conext conference* (pp. 1–12).

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis.* Cambridge,England: Cambridge University Press.

Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, *24*(4), 35–43.

Soares, C., Brazdil, P., & Kuba, P. (2004). A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, *54*(3), 195–209.

Souza, B. Feres de, Carvalho, A. C. P. L. F. de, Calvo, R., & Ishii, R. P. (2006). Multiclass svm model selection using particle swarm optimization. In *HIS '06: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems* (p. 31). Washington, DC, USA: IEEE Computer Society.

Spam assassin public mail corpus. (2002). `http://spamassassin.apache.org/publiccorpus/`. (Retrieved December 23, 2009)

S.Sonnenburg, C. B., G.Raetsch. (2006). Large scale multiple kernel learning. *The Journal of Machine Learning Research*, *7*, 1565.

Steinwart, I., & Christmann, A. (2008). *Support Vector Machines.* New York,NY: Springer Verlag.

The Times. (2010). `http://www.timesonline.co.uk`. (Last retrieved April 12, 2010)

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, *24*, 774-780. Available from `http://ci.nii.ac.jp/naid/10020952249/en/`

Vilalta, R., Giraud-Carrier, C., Brazdil, P., & Soares, C. (2004). Using meta-learning to support data mining. *International Journal of Computer Science Applications*, *1*(1), 31–45.

Zhang, X., X.L. Chen, & He, Z. (2010). An ACO-based algorithm for parameter

optimization of support vector machines. *Expert Systems with Applications*.

List of Abbreviations

AIM AOL Instant Messenger

ANOVA Analysis of Variance

AOL America On Line

DNA Deoxyribonucleic Acid

HTTP Hypertext Transfer Protocol

ML Machine Learning

POP Post Office Protocol

RMSE Root Mean Squared Error

SVM Support Vector Machine

SVR Support Vector Regression

Appendix A

Effectiveness of SVR Parameters for *N-gram* SVM Optimization

Table A.1: *Effect of SVR Parameters for N-gram SVM Optimization using the Proposed Algorithm*

SVRCost	**gamma (γ)**	**RMSE**
400	0.05	8.50992
400	0.1	5.555247
400	0.15	4.053595
400	0.2	3.346517
400	0.25	2.86574
400	0.3	2.550911
400	0.35	2.36238
400	0.4	2.24941
400	0.45	2.124799
400	0.5	2.042941
400	0.55	1.953107
400	0.6	1.862597
400	0.65	1.7832
400	0.7	1.697626
400	0.75	1.629563
400	0.8	1.567929
400	0.85	1.507324
400	0.9	1.465667
400	0.95	1.407006
450	**0.05**	**1.890609**
450	0.1	1.708259
450	0.15	1.500529
450	0.2	1.398793
450	0.25	1.306556
450	0.3	1.250609
450	0.35	1.227805
450	0.4	1.224906
450	0.45	1.21414
450	0.5	1.201731
450	0.55	1.181248

Continued on next page

Table A.1 – continued from previous page

SVRCost	gamma (γ)	RMSE
450	0.6	1.160545
450	0.65	1.129874
450	0.7	1.103122
450	0.75	1.080767
450	0.8	1.056625
450	0.85	1.0345
450	0.9	1.015517
450	0.95	0.995293
500	**0.05**	**1.352195**
500	0.1	1.222869
500	0.15	1.098058
500	0.2	1.03095
500	0.25	0.97668
500	0.3	0.944186
500	0.35	0.936754
500	0.4	0.937118
500	0.45	0.941975
500	0.5	0.93572
500	0.55	0.923687
500	0.6	0.912009
500	0.65	0.892681
500	0.7	0.877804
500	0.75	0.864118
500	0.8	0.847118
500	0.85	0.838921
500	0.9	0.822773
500	**0.95**	**0.810938**
550	0.05	1.105481
550	0.1	0.995947
550	0.15	0.906221
550	0.2	0.854153

Continued on next page

Table A.1 – continued from previous page

SVRCost	gamma (γ)	RMSE
550	0.25	0.807167
550	0.3	0.790453
550	0.35	0.790408
550	0.4	0.792143
550	0.45	0.795618
550	0.5	0.792143
550	0.55	0.785088
550	0.6	0.772235
550	0.65	0.760269
550	0.7	0.75061
550	0.75	0.739332
550	0.8	0.728195
550	0.85	0.718893
550	0.9	0.710875
550	0.95	0.702208
600	0.05	0.956616
600	0.1	0.854081
600	0.15	0.788217
600	0.2	0.743332
600	0.25	0.70329
600	0.3	0.694138
600	0.35	0.697014
600	0.4	0.698788
600	0.45	0.702004
600	0.5	0.699129
600	0.55	0.693905
600	0.6	0.682666
600	0.65	0.673551
600	0.7	0.665763
600	0.75	0.655614
600	0.8	0.650605

Continued on next page

Table A.1 – continued from previous page

SVRCost	gamma (γ)	RMSE
600	0.85	0.641103
600	0.9	0.633649
600	0.95	0.62885
650	0.05	0.857263
650	0.1	0.757525
650	0.15	0.708737
650	0.2	0.665604
650	0.25	0.633033
650	0.3	0.624846
650	0.35	0.632159
650	0.4	0.632542
650	0.45	0.634852
650	0.5	0.632408
650	0.55	0.625832
650	0.6	0.618099
650	0.65	0.611103
650	0.7	0.603832
650	0.75	0.595498
650	0.8	0.589585
650	0.85	0.584199
650	0.9	0.578295
650	0.95	0.574557

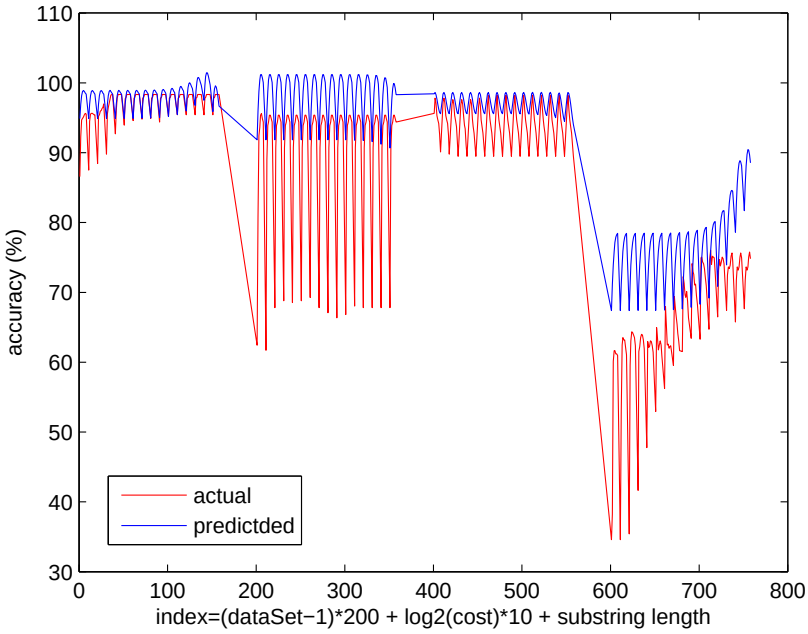


Figure A.1: *N-gram SVM Optimization using the Proposed Algorithm for 512 Parameter Combinations (SVR-Cost=450, gamma (γ)=0.05 and RMSE=1.890609), index: $0 \leq$ Spam $<$ 200 $\leq$ Reuters-21578 $<$ 400 $\leq$ Network Application Detection $<$ 600 $\leq$ e-News Categorization $<$ 800*

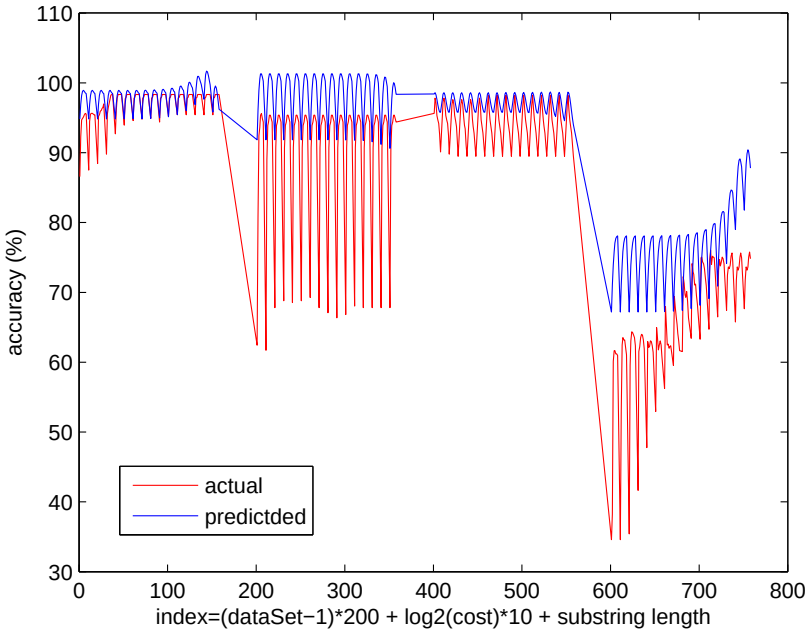
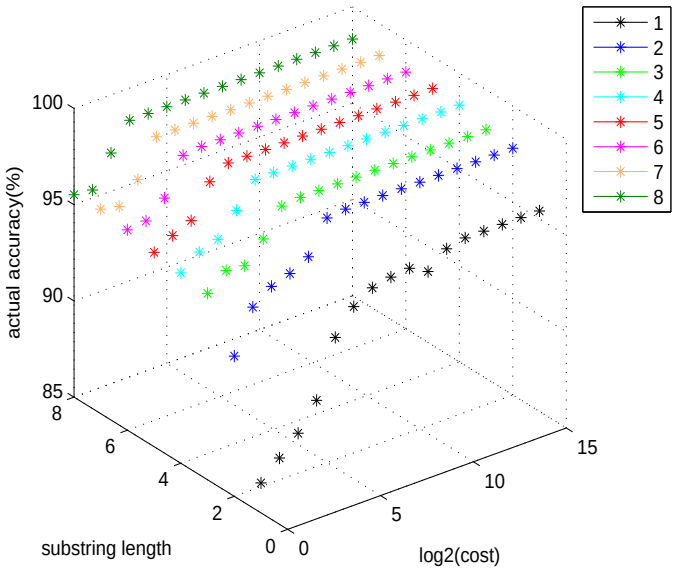


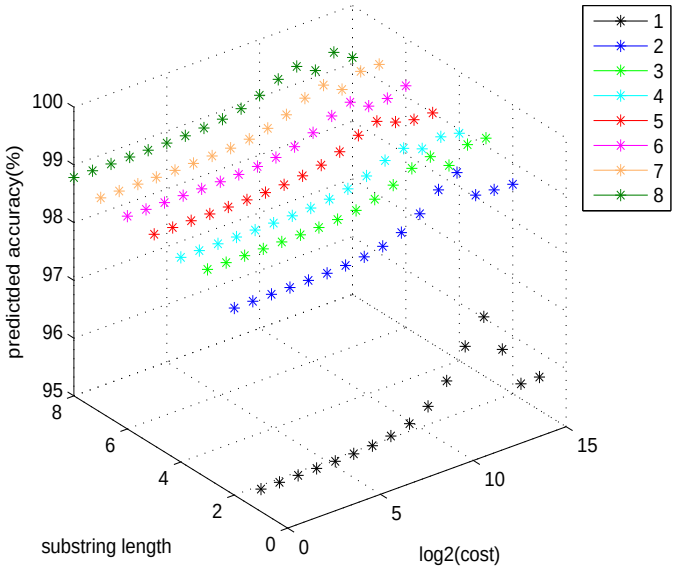
Figure A.2: *N-gram SVM Optimization using the Proposed Algorithm for 512 Parameter Combinations (SVR-Cost=500, gamma (γ)=0.05 and RMSE=1.352195), index: $0 \leq$ Spam $<$ 200 $\leq$ Reuters-21578 $<$ 400 $\leq$ Network Application Detection $<$ 600 $\leq$ e-News Categorization $<$ 800*

Appendix B

Prediction Behaviour of the Proposed Algorithm for *N-gram* SVM Optimization: Figures

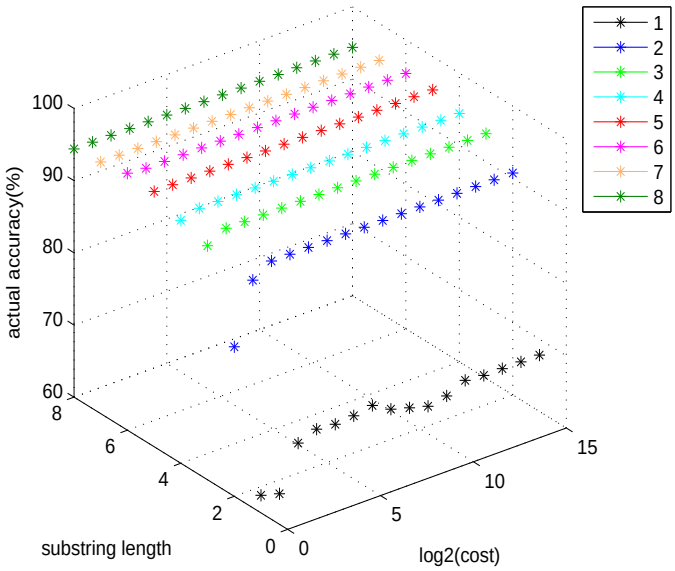


(a) Actual Parameter Behaviour: Spam (different colours represent different substring lengths)

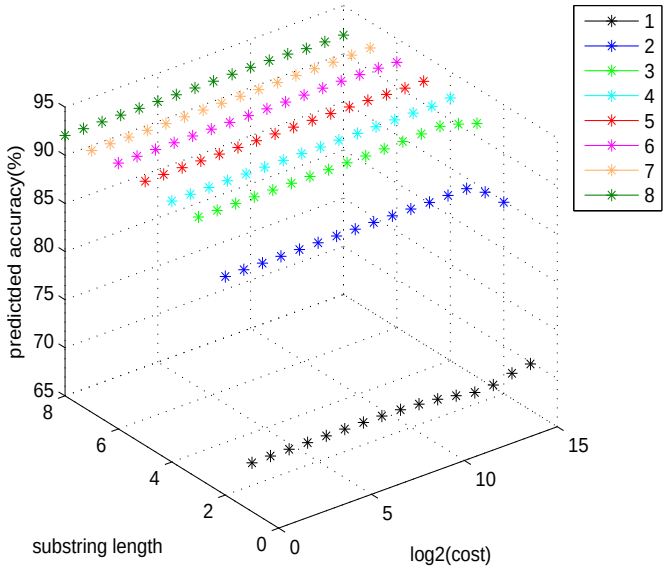


(b) Predicted Parameter Behaviour: Spam (different colours represent different substring lengths)

Figure B.1: *Prediction Behaviour of the Proposed Algorithm for N-gram SVM Optimization on Spam String Dataset (using SVR cost=500 and γ=0.95 in regression): (a) Actual parameter behaviour of n-gram SVM on Spam string dataset. (b) Predicted parameter behaviour of n-gram SVM, by the proposed algorithm on Spam string dataset.*

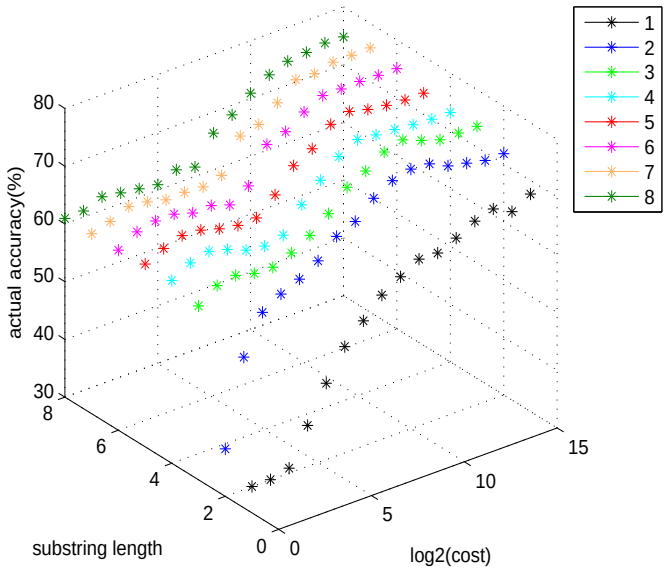


(a) Actual Parameter Behaviour: Reuters-21578 (different colours represent different substring lengths)

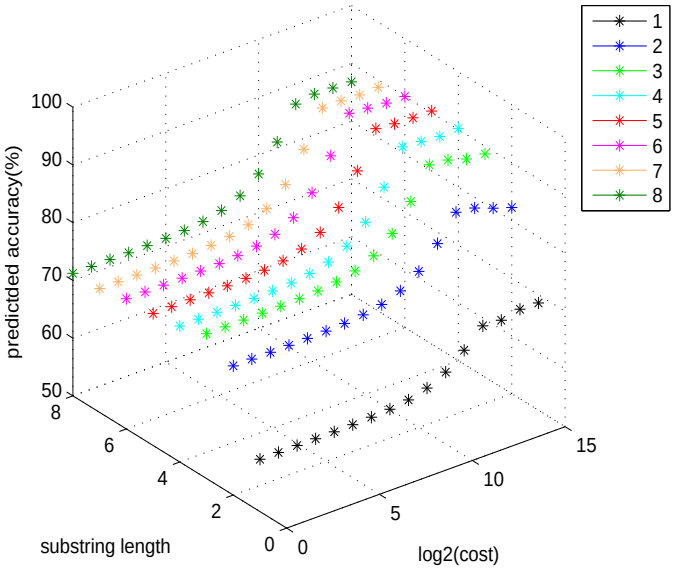


(b) Predicted Parameter Behaviour: Reuters-21578 (different colours represent different substring lengths)

Figure B.2: *Prediction Behaviour of the Proposed Algorithm for N-gram SVM Optimization on Reuters-21578 String Dataset (using SVR cost=500 and γ=0.95 in regression): (a) Actual parameter behaviour of n-gram SVM on Reuters-21578 string dataset. (b) Predicted parameter behaviour of n-gram SVM, by the proposed algorithm on Reuters-21578 string dataset. (different substring lengths are represented in different colours)*



(a) Actual Parameter Behaviour: e-News Categorization (different colours represent different substring lengths)



(b) Predicted Parameter Behaviour: e-News Categorization (different colours represent different substring lengths)

Figure B.3: *Prediction Behaviour of the Proposed Algorithm for N-gram SVM Optimization on e-News Categorization String Dataset (using SVR cost=500 and γ=0.95 in regression): (a) Actual parameter behaviour of n-gram SVM on e-News Categorization string dataset. (b) Predicted parameter behaviour of n-gram SVM, by the proposed algorithm on e-News Categorization string dataset. (different substring lengths are represented in different colours)*