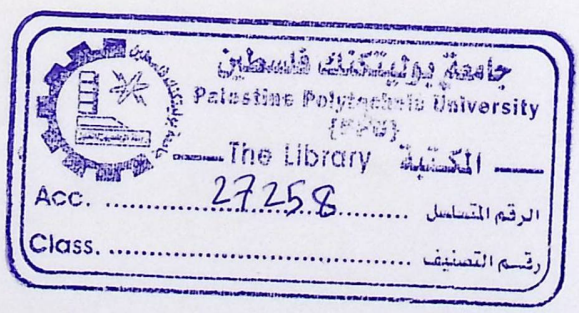
Palestine Polytechnic University
Deanship of Graduate Studies and Scientific Research
Master of informatics

Evolutionary Based Optimization of String Kernels for Support Vector Machine

Submitted by

Ruba Sultan

Thesis submitted in partial fulfillment of requirements of the degree Master of Science in Informatics
February, 2012



The undersigned hereby certify that they have read, examined and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of a thesis entitled:

Evolutionary Based Optimization of String Kernels for Support Vector Machine

Submitted by Ruba Y.S Sultan

In partial fulfillment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

Dr. Hashem Tamimi(Supervisor),
Palestine Polytechnic University
Signature:_____ Date: 19-3-2012

Dr. Yaqoub Ashhab(Co-Supervisor),
Palestine Polytechnic University
Signature:_____ Date: 15-3-2012

Dr. Mohammoud Alsaheb(Internal Examiner),
Palestine Polytechnic University
Signature:_____ Date: 26-3-2012

Dr. Rashid Jayousi(External Examiner),
AL-Quds University
Signature: R. Jayousi Date: 4/3/2012

Thesis Approved

Prof. Dr. Karim Tahboub
Dean of Graduate Studies and Scientific Research
Palestine Polytechnic University

Signature:_____ Date: 28.3.2012

Abstract

We introduce a novel approach that automatically develops a new optimized string kernel using evolutionary approaches. The new evolved kernel is used to enhance the prediction performance of Support Vector Machines (SVMs) especially in biological sequences, as it is one of the most promising classifiers in this field. The proposed approach is based on a hybrid model that combines the evolutionary algorithm with a kernel based SVM classifier. This model creates the optimized kernel from available string kernels and it optimizes the kernels and SVM parameters.

Two evolutionary approaches are examined, the Genetic Programming (GP) and the Genetic Algorithm (GA). In GP each individual represents a tree that encodes the mathematical expression of the evolved kernel. The evolved kernel could be either a combination of weighted sum of existing string kernels, or could be a mathematical expression of kernels. Many experiments with varying parameters are made to evolve the best optimized string kernel, and to optimize the kernel and SVM parameters. However, GA is used to evolve a new string kernel, either by combining some kernels, or by making a weighted combination of all string kernels.

Using two standard benchmark datasets, signal peptide and Major Histocompatibility Complex(MHC), our evolutionary optimized kernel in combination with SVM outperforms the available string kernels and produces high

classification performance.

The obtained results show that the evolved kernel gives a higher classification performance than other string kernels, and this can be clearly noticed using the evolved kernel with SVM to classify the MHC benchmark, which is known as a challenging classification model.

The evolved kernel outperforms other string kernels by nearly 10% in all performance measures, and especially when measuring the Area Under Curve (AUC).

الملخص بالعربية

استخدام الطرق التطورية في تحسين أنوية المتسلسلة لآلة المتجهات الداعمة

يهدف البحث إلى تقديم نظام فريد يعمل على تطوير نواة متسلسلة (String Kernel) جديدة باستخدام الطرق التطورية (Evolutionary Approaches). سوف تستخدم النواة الجديدة المطورة في تحسين أداء التصنيف والتنبؤ لآلة المتجهات الداعمة (Support Vector Machine) خاصة في تصنيف المتسلسلات البيولوجية.

النظام المقترح مبني على أساس آلية مركبة تجمع بين الخوارزميات التطورية و المصنفات المبنية على أساس النواه، فالنظام ينتج النواة الأمثل من أنوية متسلسلات موجودة مسبقاً، بالإضافة الى أن النظام يعمل على إيجاد المعاملات المثلى لكل نواة متسلسلة والمعاملات المثلى لأله المتجهات الداعمة.

تم إستخدام نوعين من الطرق التطورية في التجارب، هما البرمجة الجينية (Genetic Programming) والخوارزمية الجينية (Genetic Algorithm).في البرمجة الجينية، يتم تمثيل كل حل على شكل شجرة تمثل التعبير الرياضي لنواة المتسلسلة، وهنا النواةالمطورةالناتجة تكون إما مجموع متزن لانوية متسلسلات، أو نتيجة تطبيق عمليات رياضية على أنوية متسلسلات موجودة مسبقاً.أما الخوارزمية الجينية فتعمل على تطوير نواة متسلسلة جديدة إما بجمع بعض أنوية متسلسلات، أو بجمع متزن لجميع الأنوية الموجودة مسبقاً.

تم استخدام معيارين قياسيين لإختبار نواة المتسلسلة المطورة الجديدة، المعيار الأول هو إشارات الببتيد (Signal peptide) والثاني هو معقدات التوافق النسيجي الأساسية (Major Histocompatibility Complex). حيث نفذت عمليات التصنيف بإستخدام النواة المطورة مع أله المتجهات الداعمة و أعطت نتائج عالية في التصنيفتفوق نتائج التصنيف الناتجة عن تطبيق أنوية المتسلسلات المعروفة والموجودة مسبقاً، خاصة عند تطبيق النواة الجديدة في تصنيف سلاسل معقدات التوافق النسيجي الاساسية الببتيدية.

أبرزت النواة المطورة الجديدة تقدما ملحوظا تقريبا بنسبة 10% في مقاييس جودة التصنيف بشكل عام وعند قياس المساحة تحت منحنى خصائص التشغيل للمتلقي(AUC) بشكل خاص.

Declaration

I declare that the Master Thesis entitled *"Evolutionary Based Optimization of String Kernels for Support Vector Machine"* is my own original work, and hereby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Ruba Yousef Sultan

Signature: _____ Date: 27/3/2012

Statement of Permission to Use

In presenting this thesis in partial fulfillment of the requirements for the master degree in Informatics at Palestine Polytechnic University, I agree that the library shall make it available to borrowers under rules of the library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from, reproduction, or publication of this thesis may be granted by my main supervisor, or in his absence, by the Dean of Graduate Studies and Scientific Research when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Ruba Yousef Sultan

Signature: _____ Date: 27/3/2012

Dedication

To my mother, who has been a source of encouragement and inspiration to me throughout my life. To my beloved late father, GOD bless his soul. To my dear husband Bashar, and my lovely kids Bana and Obada, the source of love, sympathy and emotional support.

Acknowledgment

From the formative stages of this thesis, to the final draft, I owe an immense debt of gratitude to my supervisors, Dr. Hashem Tamimi and Dr. Yaqoub Ashhab. Their sound advice, careful guidance, continual support, patience, encouragement and assistance that motivated me to do my best. I would also like to gratefully thank the members of committee, Dr. Mahmoud Saheb, Dr. Rashid Jayousi and Dr. Radwan Tahboub for their support. Finally, I would be remiss without mentioning my family, whose extreme generosity will be remembered always.

To each of the above, I extend my deepest appreciation.

Acknowledgment

From the formative stages of this thesis, to the final draft, I owe an immense debt of gratitude to my supervisors, Dr. Hashem Tamimi and Dr. Yaqoub Ashhab. Their sound advice, careful guidance, continual support, patience, encouragement and assistance that motivated me to do my best. I would also like to gratefully thank the members of committee, Dr. Mahmoud Saheb, Dr. Rashid Jayousi and Dr. Radwan Tahboub for their support. Finally, I would be remiss without mentioning my family, whose extreme generosity will be remembered always.

To each of the above, I extend my deepest appreciation.

Table of Contents

List of Figures

List of Figures

List of Tables

List of Abbreviations

AUC	Area Under ROC Curve
EA	Evolutionary Approaches
ES	Evolutionary Strategies
GA	Genetic Algorithms
GP	Genetic programming
MA	Memetic Algorithm
MHC	Major Histocompatibility Complex
PSO	Particle Swarm Optimization
ROC	Receiver Operating Characteristics
SVM	Support Vector Machines

Chapter 1

Introduction

1.1 Problem Statement

The machine learning techniques are becoming so necessary nowadays to solve various problems in classification [48] , regression [14] and clustering [3]. Kernel-based methods [47] have been successfully applied many fields including biological problems [4]. Since these techniques are helping researchers to transform biology from a qualitative and descriptive science to a quantitative and predictive engineering science [47]. In addition, machine learning techniques are becoming necessary to biologists due to the huge profits afforded by these techniques, especially in decreasing the experiments time and costs.

Support Vector machine (SVM) is one of the well-known kernel based machine learning techniques, and one of the most extensively explored classifiers. One key success of this technique, is that it represents the data by means of a kernel function, which defines similarities between pairs of data. It takes the relationships that are implicit in the data and make them explicit, in order to make the prediction possible.

The most important design decision in SVM is the selection of the kernel

function, that defines the feature space that the data will be mapped into, using some mapping function.

Nowadays kernel methods are witnessing a major paradigm shift. Many enhancement and developments are taking place in the kernel selection and optimization problem. The problem of selecting the optimal kernel for the SVM and selecting its optimal hyperparameters is known as model selection. This task is usually done by training the classifier with different functions taken from a list of a kernel functions and set of parameters. Intuitively, single kernels may not be able to solve complex problems, therefore multiple kernels are generally used together to solve them.

Kernel-based methods prove their efficiency not only in solving numerical data problems, but also in solving string classification problems in various fields including bioinformatics [4], where string kernels are frequently used.

While many researches deal with selecting the suitable optimized combination of numerical kernels, and optimizing a single kernel parameters, no attempt has been made in tackling optimization problem for the string kernels. The aim of this thesis is to develop an new tool that automatically designs the optimized string kernel. In the automatic kernel design, the best expression of string kernels is constructed, and the optimal hyperparameters for each sub-string kernel are found using two types of the evolutionary approaches, namely the Genetic Programming (GP) and the Genetic Algorithm (GA),(see section 2.3).

Evolutionary approaches are a family of stochastic optimization techniques whose function is based on the principles of natural evolution. GP is a variant of evolutionary optimization algorithms and is a special case of GA. It uses mechanisms inspired by natural evolution to do a parallel search using a population of candidate solutions and provides a powerful and ro-

bust means of solving problems. Optimization using numerical methods like deterministic gradient-based and simplex-based search method will not serve in finding the optimal parameters values since the objective function is ill-defined and difficult to model. So, optimization using the GP and GA will fulfill the goal in this work more effectively.

Our model uses GA to find the optimal expression for string kernels. It selects the sub-kernel from a pool of available predefined string kernels. Also our model finds the optimal parameters for each sub-kernel, and optimizes the SVM regularization parameter. We have evaluated the performance of the proposed model on classifying biological sequences, especially in classifying secreted and non-secreted proteins based on predicting the presence of the signal peptide, and in classifying binding and non-binding MHC peptides.

1.2 Thesis Objectives

The key objectives of this study are:

- Evolve a new string kernel using GP approach or GA approach, by finding the optimal combination of predefined string kernels
- Study the optimization of each sub-kernel parameter(s) and the SVM parameters
- Exploit the effectiveness, simplicity, and robustness of the genetic algorithms to perform the optimization successfully.
- Apply the resulted evolved kernel in addressing two famous biological problems: the prediction of signal peptide in secreted proteins, and the binding of peptides to MHCII system.

- Demonstrate the power of the proposed evolved optimized string kernel, and compare it with the available single ones.

1.3 Thesis Organization

This thesis is arranged as follows: Chapter 1 introduced the problem definition, and the goals of the work. Chapter 2 describes the theories and basic concepts that are needed to understand the rest of the thesis. Chapter 3 exploits our methodology and implementation issues related to SVM, kernels, and GP. Chapter 4 demonstrates experiments and results achieved by the work, and results analyses. Chapter 5 concludes the work and propose some new direction for the future work.

Chapter 2

Background

This chapter gives a theoretic basis needed for understanding the rest of the thesis. The first section explains the kernel methods, kernel characteristics, kernel types, kernel matrices, and kernel operations. A description for different string kernels that is used in our model is also given in first section. The sections that follows explain some techniques in the field of machine learning that is used in our model like SVM and evolutionary approaches, with emphasis on GP and GA. A quick overview of the performance measures used in the experiments, is also introduced.

2.1 Kernel Methods

Linear classifiers in general can not handle classification problems of non linearly separable data, so this kind of problems are usually classified using non-linear classifiers. Even of the fact that non-linear classifiers provide better accuracy than a linear ones, linear classifiers still have many advantages, like the simple training algorithms that scale well with the number of examples.

One way to get benefit of the advantages of linear classifiers, when solving

non-linear problems, is to map the data into a richer mathematical space where the data become linearly separable and then use a linear classifier in this space.

Kernel methods [47] are a class of machine learning techniques that uses mathematical kernel functions to implicitly map the data to higher dimensional space. The kernel function returns the value of the dot product of two arguments projected into the space induced by the kernel function, since the dot product is a measure of similarity between arguments [16].

Kernel methods produce a kernel matrix by comparing all data instances with each other, which is the basis for several learning algorithms that can be written in terms of dot products, but the kernel trick [57] [47] makes it possible to implicitly calculate the kernel matrix. More details about the fundamental properties of kernels is shown in Section 2.1.2.

Kernel methods map the data into a higher dimensional space to make it suitable for processing. It can be then treated by linear algebra, geometry and statistic algorithms to mine patterns from data. Any kernel method consists of two parts: a module that performs the mapping into the feature space, and a learning algorithm designed to discover linear patterns in that high dimensional space. Figure 2.1 illustrates the mapping to higher dimensional space. Linear kernel can be viewed as a natural inner product between elements of the input vector. Linear kernel corresponds to running the original algorithm in the input space itself.

Many complex kernels can be created from simpler ones in a number of different ways. Kernels that correspond to infinite dimensional feature spaces can even be constructed at the cost of only a few extra operations in the kernel evaluations.

Kernel methods show their modularity in the reusability of the learning

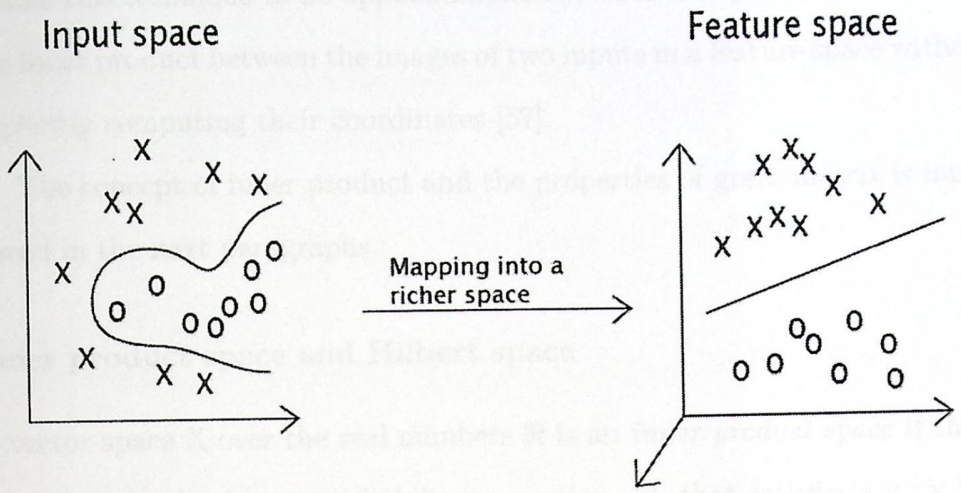


Figure 2.1: Mapping data items into a higher dimensional feature space [57].

algorithm. The same algorithm can be used with any kernel and for any data domain. The kernel content is data specific, but can be combined with different algorithms to solve the full range of tasks. This brings a very natural and elegant approach to learning systems design, where modules are combined together to obtain complex learning systems. The data is processed using a kernel to create a kernel matrix, which in turn is processed by a data mining algorithm to produce a general function that would be used to process unseen examples. Kernel functions provide a principled and powerful way of detecting nonlinear relations using well-understood linear algorithms.

2.1.1 Inner Product and Matrix Properties

Fundamental properties of kernels will be discussed here. Given a kernel function and a training set, a matrix can be formed, known as Gram matrix which is the matrix containing the evaluation of the kernel function on each pairs of data points. Data can be mapped into a high-dimensional feature space, where we can then classify the data linearly in that space. Kernels

enable this technique to be applied implicitly, since it is possible to evaluate the inner product between the images of two inputs in a feature space without explicitly computing their coordinates [57].

The concept of inner product and the properties of gram matrix is introduced in the next paragraphs.

Inner product space and Hilbert space

A vector space X over the real numbers $\Re$ is an *inner product space* if there exists a real-valued symmetric bilinear map$\langle \, . \, , \, . \, \rangle$, that satisfies $\langle \, \mathrm{x} \, , \, \mathrm{x} \, \rangle \geq 0$ [57].

A Hilbert space F is an inner product space with additional properties that is separable and complete. [57].

Symmetric matrices

A matrix A is symmetric if $A^T = A$, that is the (i, j) entry equals the (j, i) entry for all i and j. For symmetric matrices the eigenvectors corresponding to distinct eigenvalues are orthogonal [57].

Positive semi-definite matrix

A matrix A is a positive semi-definite if and only if $A = B^T B$ for some real matrix B. A useful characterization of this matrix that all its principal minors are positive semi definite. The determinant $det(A)$ of a square matrix A is the product of its eigenvalues. Hence for a positive definite matrix the determinant will be strictly positive [57].

2.1.2 Kernels Characteristics

The kernel function computes the inner product of the samples under an embedding ϕ of the two data points

$$k(x, z) = \langle \phi(x), \phi(z) \rangle \tag{2.1}$$

where ϕ is the mapping function. The kernel function implicitly defines a feature space that we do not need to construct explicitly. There is a general characteristics that a candidate function possess to be a kernel, so to verify that a function is a kernel we can search for these characteristics instead of constructing a feature space for which the function corresponds to first performing the feature mapping then computing the inner product between them. This will provide one of the theoretical tools needed to create new kernels, and combine old kernels to create new ones. One of the key feature of a kernel function is the positive semi-definiteness [57].

Finitely positive semi-definite function

A function

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

satisfies the finitely positive semi-definite property if it is a symmetric function for which the matrices formed by restriction to any finite subset of the space X are positive semi-definite. This property characterizes the kernel since the function k which is either continuous or has a finite domain, can be decomposed into a feature map ϕ into a Hilbert space F applied to both its arguments followed by the evaluation of the inner product in F if and only if it satisfies the finitely positive semi-definite property.

Given a function k, as in Equation 2.1, that satisfies the finitely posi-

tive semi-definite property we will refer to the corresponding space F_k as its Reproducing Kernel Hilbert Space (RKHS), Any kernel can be used to construct a Hilbert space in which the reproducing property holds. Then it is fairly straightforward to see that if a symmetric function $k(.,.)$ satisfies the reproducing property in a Hilbert space F of functions

$$\langle k(x,.), f(.)\rangle_F = f(.)\ , for\ f \in F \tag{2.2}$$

then k satisfies the finitely positive semi-definite property [57]. Since the validity of kernel is already achieved with the RKHS construction, then Mercers theorem (that is usually used to construct a feature space for a valid kernel) is not actually needed in kernel construction [57].

Kernel Trick

Kernel trick denotes for the process of calculating the dot product in a feature space $\langle \phi(x), \phi(z)\rangle$ without the need of performing the the mapping $\phi(x)$. Instead, it is sufficient to calculate this product directly in the input space by computing the squared dot product between the samples $\langle x, z\rangle^2$ [61].

So, the kernel function implicitly defines a feature space that we do not need to construct explicitly. The following example illustrates the idea of the kernel trick:
Let $x \in \mathbb{R}^2$ i.e., x $= [x_1 x_2]^T$, and if we choose $\phi(x) = [x_1^2\ \sqrt{2}x_1x_2\ x_1^2]^T$ i.e., there is an $\mathbb{R}^2$ to $\mathbb{R}^3$ mapping, then the dot product

$$\begin{aligned}\left\langle \phi^T(x), \phi(z)\right\rangle &= [x_1^2\ \sqrt{2}x_1x_2\ x_1^2]^T[z_1^2\ \sqrt{2}z_1z_2\ z_1^2] \\ &= [x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_1^2z_1^2] \\ &= \langle x, z\rangle^2\end{aligned}$$

So $K(x,z) = \langle x,z\rangle^2 = \langle \phi^T(x), \phi(z)\rangle$

Interestingly, there are also other mappings accomplish the same task as $\langle x,z\rangle^2$ such as $\mathbb{R}^2 \to \mathbb{R}^3$ mapping given by $\phi(x) = [x_1^2 - x_2^2\ 2x_1x_2\ x_1^2 + x_2^2]$ or $\mathbb{R}^2 \to \mathbb{R}^4$ mapping given by $\phi(x) = [x_1^2\ x_1x_2\ x_1x_2\ x_2^2]$

2.1.3 Kernel Matrix

Kernel matrix can be thought as information bottleneck, it is the core ingredient in the theory of kernel methods. It contains all the information available in order to perform the learning step, with the sole exception of the output labels in the case of supervised learning. The kernel or Gram matrix K $=(K_{ij})_{i,j=1}^{\ell}$, with entries $K_{ij} = k(x_i, x_j)$, for i,j=1, . . . , ℓ , given a training set S= $\{x1, ..., x_\ell\}$.

The kernel is a container that has all the information needed to the learning machine as it gives a knowledge about the relative positions of the inputs in the feature space. It defines the similarity measure between two data points, it gives a priori probability of the inputs being in the same class minus the priori probability of their being in other class.

It is known that only through the kernel matrix, the learning algorithm obtains information about the choice of feature space or model, and indeed the training data itself. Through the kernel matrix the learning algorithm receives information about the feature space and input data, so it plays a central role both in the derivation of generalization bounds and in their evaluation in practical applications.

The kernel matrix is not only the central concept in the design and analysis of kernel machines, it can also the central data structure in their implementation. The kernel matrix acts as an interface between the data input module and the learning algorithms, its properties affect every part of learn-

ing system from computation,generalization analysis to the implementation.

One issue regarding implementation is the memory constrains, that it is may not possible to store the full matrix in memory for very large dataset, so in this case it is necessary to recompute the kernel function as needed. This need to considered in the implementation details.

The same properties of a kernel function is kept for all kinds of inputs, real vectors, strings, discrete structures, images, time series, etc. So the kernel matrix corresponding to any finite training set, that is positive semi-definite, it computes the inner product after projecting pairs of input into some feature space. Figure 2.2 shows the embedding that the objects are mapped into feature vector by a mapping function ϕ [57] [61].

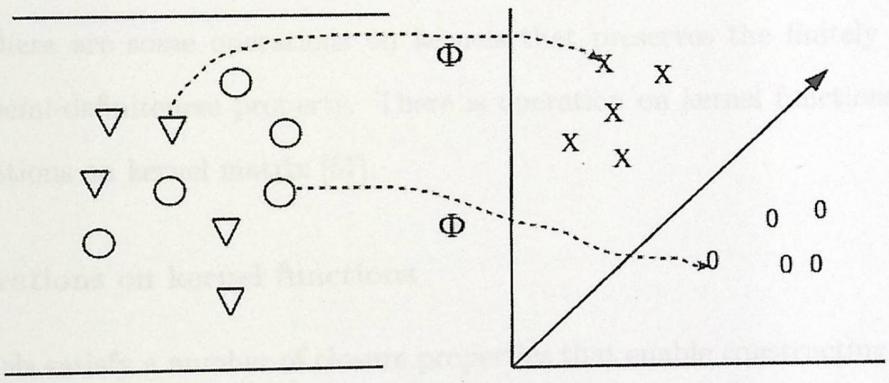


Figure 2.2: A mapping function ϕ used to map data items

2.1.4 Kernel Selection and Construction

Selecting a kernel is ideally done based on a prior knowledge of the problem domain and restrict the learning to the task of selecting the particular pattern function in the feature space defined by the chosen kernel. Unfortunately, it is not always possible to make the right choice of kernel a priori, however one should choose a family of kernels defined in a way that both reflects the

researcher prior expectations and leaves open the choice of the particular kernel that will be used.

In fact there is a wide range of valid kernels: some are given in closed form; others can only be computed by means of a recursion or other algorithm; in some cases the actual feature mapping corresponding to a given kernel function is not known. Selecting the best kernel function from among large range of possibilities becomes the most critical stage in kernel-based algorithms.

One of the benefits that can be obtained from the characterization of kernel function and kernel matrices that mentioned above, is to decide whether a given candidate is a valid kernel, also this can be used to justify a series of rules for combining simple kernels to produce more complex ones.

There are some operations on kernels that preserves the finitely positive semi-definiteness property. There is operation on kernel functions and operations on kernel matrix [57].

Operations on kernel functions

Kernels satisfy a number of closure properties that enable constructing more complicated kernels from simple building blocks, these properties can be viewed in the following proposition [57].

Proposition 1 ***(Closure properties)*** *Let k1 and k2 be kernels over* $\mathrm{X} \times \mathrm{X}$, $\mathrm{X} \in \mathbb{R}^{\mathrm{n}}$, $\mathrm{a} \in \mathbb{R}^{+}$, $\mathrm{f}(.)$ *a real-valued function on* X,$\phi : X \to \mathbb{R}^N$ *with* k_3 *a kernel over* $\mathbb{R}^N \times \mathbb{R}^N$, *and* B *a symmetric positive semi-definite* $n \times n$ *matrix. Then the following functions are kernels:*

1. *Summation of two kernel:*

$$k(x, z) = k_1(x, z) + k_2(x, z) \tag{2.3}$$

2. *Multiplying a kernel by constant:*

$$k(x, z) = ak_1(x, z) \tag{2.4}$$

3. *Product of two kernels:*

$$k(x, z) = k_1(x, z)k_2(x, z) \tag{2.5}$$

4. *Multiply a function of x by a function of z:*

$$k(x, z) = f(x)f(z) \tag{2.6}$$

5. *Calculating a kernel for the mapped x,z:*

$$k(x, z) = k_3(\phi(x), \phi(z)) \tag{2.7}$$

6. *Product of x^T by z with any positive semidefinite matrix:*

$$k(x, z) = x^T \mathrm{B} z \tag{2.8}$$

and let p(x) is polynomial with positive coefficients. Then the following functions are also kernels:

1. *Product of the kernel and polynomial*

$$k(x, z) = p(k_1(x, z)) \tag{2.9}$$

2. *Exponential of the kernel*

$$k(x, z) = \exp(k_1(x, z) \tag{2.10}$$

3. *Gaussian kernel:*

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \tag{2.11}$$

Operation on kernel matrices

There are many operations on kernel matrices that preserve symmetry and positive semi-definiteness property. Simple transformation such as adding a constant to all entries of the matrix, another simple operation is adding constant to the diagonal.

Another operation that can be performed is centering the data in the feature space, by minimizing the trace of the kernel matrix or equally minimize the sum of its eigenvalues. Subspace projection, whitening and sculpting the feature space are other operations that can be performed on kernel matrices [57].

2.1.5 Numerical Kernels

Kernel functions can be designed for vectorial inputs, objects and structures as strings, graphs, text documents. here, a description of numerical kernels is introduced.

The input for these kernels are a numerical vectors, we introduces two famous examples of numerical kernels: polynomial and Gaussian kernels.

The derived polynomial kernel for a kernel k_1 is defined as

$$k(x,z) = p(k_1(x,z)) \tag{2.12}$$

where $p(.)$ is any polynomial with positive coefficients. Frequently, it also refers to the special case:

$$k_d(x,z) = (\langle x,z\rangle + B)^d \tag{2.13}$$

defined over a vector space X of dimension n, where B and d are scalar parameters. The dimension of the feature space for the polynomial kernel is $\begin{pmatrix} n+d \\ d \end{pmatrix}$.

One of the most widely used used kernels is the Gaussian kernel, it defined by

$$k(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) \quad for\ \sigma > 0 \tag{2.14}$$

The parameter σ controls the flexibility of the kernel as the degree d in the polynomial kernel. Small values of σ correspond to large values of d because they allow classifiers to fit any labels, hence risking overfitting. In such cases the kernel matrix becomes close to the identity matrix. On the other hand, large values of σ gradually reduce the kernel to a constant function, making it impossible to learn any non-trivial classifier.

String Kernels

The inputs for these kernels are free text and symbol strings of varying lengths, such as bioinformatics data, that is used to represent proteins as sequences of 20 amino acids(`D,E,H,K,R,N,Q,S,T,Y,A,L,P,M,G,V,I,F,W,C`), genomic DNA as sequences of 4 different nucleotides(`A,T,C,G`). String ker-

nels enables the kernel methods in general to operate in a domain that traditionally has belonged to syntactical pattern recognition, and it provides a bridge between that field and statistical pattern analysis. String kernels embed a two sequences in a high-dimensional space such that their relative distance in that space reflects their similarity and the inner product between their images can be computed efficiently [57] . It is important to know the similarity notion that should be reflected in the embedding, or the significant features of the sequences that should be taken in consideration in this embedding.

A meaningful similarity notion in biological applications is counting substrings or subsequences that the two strings have in common. since it results give a good indication about the functional similarity that the bioinformatics researchers would like to capture. An example of a simple string kernel that uses the natural for comparing two strings, that is to count how many contiguous substrings of length p they have in common. This kernel is called *"Spectrum Kernel"* [26]. A brief description of this kernel and other string kernels will be introduced in the following subsections.

Spectrum Kernel

The spectrum of order p (or p-spectrum) of a sequence s is defines as the histogram of frequencies of all its contiguous substrings of length p [26]. An important information about the similarity between sequences can be obtained from comparing the p-spectra of them in applications where contiguity plays an important role, then the kernel can be defines as the inner product of their p-spectra [57] [26]. The feature space F associated with the

p-spectrum kernel is indexed by $I = \sum p$, with the embedding given by

$$\phi_u^p(s) = |\{(v_1, v_2) : s = v_1 u v_2\}| \ , u \in \sum p. \tag{2.15}$$

The associated kernel is defined as

$$k_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \sum p} \phi_u^p(s) \phi_u^p(t) \tag{2.16}$$

Weighted spectrum kernel is a spectrum kernel for 1 to $p - mers$, where each $p - mer$ length is weighted by some coefficient β_k [51]. This kernel is given by:

$$k(s,t) = \sum_{p=1}^{P} \beta_p \phi_p(s) \phi_p(t) \tag{2.17}$$

An example of this kernel: p-spectrum kernel: p=3, can be shown in table 2.1 and table 2.2

Table 2.1: Two protein sequences used in the example of spectrum kernel

seq-name	seq
s	AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
t	TACCTAATTATGAAATTAAATTTCAGTGTGCTGATGGAAACGGAGAAGTC

Table 2.2: The number of occurrences of the 3-mers in the two sequences in the spectrum kernel example : in this table each 3-mers is listed, and then the number of its occurrences in each sequence is counted

3-mer	AAA	AAC	...	CCA	CCC	...	TTT
No in s	2	4	...	1	0	...	5
No in t	3	1	...	0	0	...	1

$$k(s,t) = < [2\ 4\ ...\ 1\ 0\ ...\ 3], [3\ 1\ ...\ 0\ 0\ ...\ 1] >$$
$$= 2 \times 3 + 4 \times 1 + ... + 1 \times 0 + 0 \times 0 + ... + 3 \times 1$$

Fixed-Degree Kernel

This kernel can be seen as special case of spectrum kernel, while it considers substrings that have a given fixed length p [57]. Its feature space is indexed by $\sum p$, with the embedding given by

$$\phi_u^p(s) = |\{(i : u = s(i)\}|, u \in \sum p. \tag{2.18}$$

Then the fixed degree string kernel is given by:

$$k_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \sum p} \phi_u^p(s) \phi_u^p(t) \tag{2.19}$$

An example on the Fixed degree kernel: d=3, can be shown in table 2.3 and table 2.4.

Table 2.3: Two protein sequences used in the example of Fixed degree kernel

seq-name	seq
s	ACAGATAAGTAACCCTTTGCCTGCGTA
t	TGAGATAATTATTCTGGTGCCAAACTG

Table 2.4: The number of matches of the 3-mers in the two sequences, is listed for the fixed degree kernel example: in this table each 3-mers is listed, and then the number of matches in the two sequences is counted

3-mer	ACA	CAG	...	TGC	GCC	...	GTA
Match	0	0	...	0	1	...	0

$$k(s,t) = 0 + 0 + ... + 0 + 1 + ... + 0$$

Polynomial String Kernel

This kernel computes a variant of the polynomial kernel on strings [51]. It is computed as

Fixed-Degree Kernel

This kernel can be seen as special case of spectrum kernel, while it considers substrings that have a given fixed length p [57]. Its feature space is indexed by $\sum p$, with the embedding given by

$$\phi_u^p(s) = |\{(i : u = s(i)\}|, u \in \sum p. \tag{2.18}$$

Then the fixed degree string kernel is given by:

$$k_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \sum p} \phi_u^p(s) \phi_u^p(t) \tag{2.19}$$

An example on the Fixed degree kernel: d=3, can be shown in table 2.3 and table 2.4.

Table 2.3: Two protein sequences used in the example of Fixed degree kernel

seq-name	seq
s	ACAGATAAGTAACCCTTTGCCTGCGTA
t	TGAGATAATTATTCTGGTGCCAAACTG

Table 2.4: The number of matches of the 3-mers in the two sequences, is listed for the fixed degree kernel example: in this table each 3-mers is listed, and then the number of matches in the two sequences is counted

3-mer	ACA	CAG	...	TGC	GCC	...	GTA
Match	0	0	...	0	1	...	0

$$k(s,t) = 0 + 0 + ... + 0 + 1 + ... + 0$$

Polynomial String Kernel

This kernel computes a variant of the polynomial kernel on strings [51]. It is computed as

$$k(\mathbf{s},\mathbf{t}) = (\sum_{i=0}^{L} I(s_i = t_i) + c)^d \tag{2.20}$$

where I is the indicator function which evaluates to 1 if its argument is true and to 0 otherwise [51].

Since polynomial kernel is originally defined on real-valued inputs, it cannot directly be applied to discrete data, like DNA and Protein. So a common technique that is used to map the alphabets into a binary representation. The following is an example on DNA sequence: let s $\in \{A, C, G, T\}^N$ then it is represented as

$$\begin{aligned} s' = (&I(s_1 = A), I(s_1 = C), I(s_1 = G), I(s_1 = T), \\ &I(s_2 = A), I(s_2 = C), I(s_2 = G), I(s_2 = T), \ldots, \\ &I(s_N = A), I(s_N = C), I(s_N = G), I(s_N = T))^{\mathrm{T}} \end{aligned}$$

This kernel takes all correlations of matches $I(s_i = t_i)$ up to order d into account. The features used for learning are position-dependent. They carry local and global information about the sequence, as, for instance, any position is combined with any other position to form a feature in the kernel feature space generated by raising the scalar product to the power of d [43]. An example on this kernel: Polynomial kernel: d=4, c=5, can be shown in table 2.5 and table 2.6.

Table 2.5: Two protein sequences used in the example of Polynomial kernel

seq-name	seq
s	ACAGATAAGTAACCCTTTGCCTGCGTA
t	TGAGATAATTATTCTGGTGCCAAACTG

$$k(\mathbf{s},\mathbf{t}) = (\sum_{i=0}^{L} I(s_i = t_i) + c)^d \tag{2.20}$$

where I is the indicator function which evaluates to 1 if its argument is true and to 0 otherwise [51].

Since polynomial kernel is originally defined on real-valued inputs, it cannot directly be applied to discrete data, like DNA and Protein. So a common technique that is used to map the alphabets into a binary representation. The following is an example on DNA sequence: let s $\in \{A, C, G, T\}^N$ then it is represented as

$$\begin{aligned} s' = (&I(s_1 = A), I(s_1 = C), I(s_1 = G), I(s_1 = T), \\ &I(s_2 = A), I(s_2 = C), I(s_2 = G), I(s_2 = T), \ldots, \\ &I(s_N = A), I(s_N = C), I(s_N = G), I(s_N = T))^{\mathrm{T}} \end{aligned}$$

This kernel takes all correlations of matches $I(s_i = t_i)$ up to order d into account. The features used for learning are position-dependent. They carry local and global information about the sequence, as, for instance, any position is combined with any other position to form a feature in the kernel feature space generated by raising the scalar product to the power of d [43]. An example on this kernel: Polynomial kernel: d=4, c=5, can be shown in table 2.5 and table 2.6.

Table 2.5: Two protein sequences used in the example of Polynomial kernel

seq-name	seq
s	ACAGATAAGTAACCCTTTGCCTGCGTA
t	TGAGATAATTATTCTGGTGCCAAACTG

Table 2.6: An indication of the match for each amino acid in the two sequences for the polynomial kernel example: in this table each amino acid in the sequence is listed, and then the match=1 when the two corresponding amino acids are the same, and match=0 in the opposite case.

	A	C	A	G	A	...	A
Match	0	0	1	1	1	...	0

$$k(s,t) = ((0+0+1+1+1...+0)+5)^4$$

Locality Improved Kernel

The locality-improved kernel uses a small sliding window to scan the input sequence and counts matching nucleotides in every window [65]. All these counts are raised to the power of d_1 (see equation 2.21)and then are added up. At last, the sum is taken to the power of d_2(see equation 2.21). Here, d_1 and d_2 are user-specified parameters [29]. In this kernel, at each sequence position, a comparison of the two sequences is performed locally, within a small window (win) of length 2ℓ+1 around that position. Again, the matching nucleotides is counted, this time multiplied with weights w increasing from the boundaries to the center of the window. The resulting weighted counts are taken to the power d_1^{th}. d_1 reflects the order of local correlations within the window is expected to be of importance.

$$win_p(s,t) = \left(\sum_{j=-\ell}^{+\ell} w_j match_{p+j}(s,t) \right)^{d_1} \tag{2.21}$$

Here, $match_{p+j}(s,t)$ is 1 for matching nucleotides at position p+j and 0 otherwise. The window scores computed with win_p are summed over the whole length of the sequence. Correlations between up to d_2 windows are

Table 2.6: An indication of the match for each amino acid of d_2. quences for the polynomial kernel example: in this table each amino the sequence is listed, and then the match=1 when the two corresponding amino acids are the same, and match=0 in the opposite case.

	A	C	A	G	A	...	A
Match	0	0	1	1	1	...	0

(2.22)

For the polynomial kernel, each window score is a kernel that includes all monomial features of degree d_1. The monomials are

$$k(s,t) = ((0+0+1+1+1+1...+0)^{d_1}$$

weighted according to the representation of correlations of sequence positions that use d_2 intra-window monomials. Thus, distant correlations only values $d_2 > 1$. Intuitively, it is clear that this

Locality Improved Kernel

The locality-improved kernel uses a small sliding window kernel function, since it corresponds to the application of sequence and counts matching nucleotides in every polynomial map with degree d_2 to an intermediate space that is counts are raised to the power of d_1 in the feature space of a weighted polynomial map with degree d_1 on the At last, the sum is taken to the power of d_1 (see equation [65]. An example of the Locality Improved kernel: win-length=3, are user-specified parameters [29]. In this can be shown in table 2.7 and table 2.8.

comparison of the two sequences used in the example of Locality improved

seq-name	seq
s	ACAGATAAGTAACCCTTTGCCTGCGTA
t	TGAGATAATTATTCTGGTGCCAAACTG

(win) of length $2\ell+1$ around that position is counted, this time multiplied by a weight that depends on the distance to the center of the window. The result is raised to the power d_1^{th}. d_1 reflects the order of correlations expected in the locality improved kernel: In this table, from window-1 to window-N is viewed, then the matches in 1-mers, 2-mers, 3-mers, is counted among the two sequences.

	win_1	win_2	win_3	win_4	...	win_N
match1	0	0	1	1	...	0
match2	0	1	1	1	...	1
match3	1	1	1	1	...	0

Here, $\ldots$ otherwise

(2.24)

taken into account by raising the resulting sum to the power of d_2.

$$k(s,t) = \left(\sum_{p=1}^{N} win_p(s,t) \right)^{d_2} \tag{2.22}$$

This kernel is similar to the polynomial kernel, each window score is a kernel that induces a set of monomial features of degree d_1. The monomials are weighted in order to strengthen the representation of correlations of sequence positions that are close d_2 intra-window monomials. Thus, distant correlations are taken into account by values $d_2 > 1$. Intuitively, it is clear that this function is a valid kernel function, since it corresponds to the application of a weighted polynomial map with degree d_2 to an intermediate space that is defined as feature space of a weighted polynomial map with degree d_1 on the input space. [65] An example of the Locality Improved kernel: win-length=3, d_1=2, d_2=3, can be shown in table 2.7 and table 2.8.

Table 2.7: Two protein sequences used in the example of Locality improved kernel

seq-name	seq
s	ACAGATAAGTAACCCTTTGCCTGCGTA
t	TGAGATAATTATTCTGGTGCCAAACTG

Table 2.8: The number of matches of different length mers in different windowing, in the locality improved kernel: In this table, from window-1 to window-N is viewed, then the matches in 1-mers, 2-mers, 3-mers, is counted among the two sequences.

	win_1	win_2	win_3	win_4	...	win_N
match1	0	0	1	1	...	0
match2	0	1	1	1	...	1
match3	1	1	1	1	...	0

$$win_1 = (0 \times w_1 + 0 \times w_2 + 1 \times w_3)^2$$
$$win_2 = (0 \times w_1 + 1 \times w_2 + 1 \times w_3)^2$$
$$....$$
$$win_N = (0 \times w_1 + 1 \times w_2 + 1 \times w_3)^2$$
$$k(s,t) = (win_1 + win_2 + ... + win_N)^3$$

Local Alignment Kernel

Local alignments provides a powerful technique for detecting similarity between sequences, using the optimal local alignment via the Smith-Waterman algorithm [50] and it's efficient PSI-BLAST approximations [2]. So Local Alignment Kernel comprises several sub-kernels based on the Smith-Waterman algorithm [59]. On a protein homology detection problem, this approach was found to significantly outperform scores based solely on optimal alignments [44]. Local Alignment Kernels are convolution kernels [17] consisting of a number of simple sub-kernels:

$$k_1 \cdot k_2 \cdot .... \cdot k_3(s,t) = \sum_{s=s_1...s_p, t=t_1...t_p} k_1(s_1,t_1) \cdot ... \cdot k_p(s_p,t_p) \qquad (2.23)$$

Where the components $k1...kp$ consists of three different kernels [45]:

- A constant kernel k_{const}
- a kernel for measuring the difference between aligned letters k_{align}
- a kernel for penalizing gaps k_{gap}

$$k_{const}(s,t) = 1$$
$$k_{align}(s,t) = \begin{cases} 0, if\ |s| \neq 1 or\ |t| \neq 1 \\ e^{\beta * S(s,t) otherwise} \end{cases} \qquad (2.24)$$
$$k_{gap}(s,t) = e^{\beta(g(|s|)+g(|t|))}$$

where s and t are the amino acid sequences, $S(s,t)$ the Smith-Waterman score, $g(.)$ the gap penalty function, and β a scaling parameter to adjust importance of gaps and sub-optimal alignments. The Smith-Waterman score $SW_{S,g(\pi)}$ is calculated as:

$$S_{S,g(\pi)} = \sum_{i=1}^{|\pi|} S(s_{\pi_1}(i), t_{\pi_2}(i)) - \sum_{i=1}^{|\pi|-1} [g(\pi_1(i+1) - \pi_1(i)) + g(\pi_2(i+1) - \pi_2(i+1) - \pi_2(i))] \tag{2.25}$$

Where π is the alignment between two sequences s and t, $S(.)$ denotes for the substitution matrix and $g(.)$ a gap penalty function. The component sub-kernels are combined by convolution to represent a kernel for an alignment of length n. The Local Alignment score is the sum over all possible alignments in the sequence:

$$\begin{aligned} k_{(n)}(s,t) &= k_{const} \cdot (k_{align} \cdot k_{gap})^{(n-1)} \cdot k_{align} \cdot k_{const} \\ k_{LA}(s,t) &= \sum_{i=0}^{N} k_i(s,t) \end{aligned} \tag{2.26}$$

where N is the number of all possible alignments [45]. An example on the local alignment kernel can be shown in table 2.9.

Table 2.9: Two protein sequences used in the example of Local Alignment kernel: the two sequences are aligned, then the matches and difference is shown.

seq-name	seq
s	ACAGATAA----ATTC
	|...||||....||||
t	A---AGAAATTGATTC

$$S_{S,g}(\pi_1) = S(A,A) + S(A,A) + S(T,G) + 3 \times S(A,A) + 2 \times S(T,T) + S(C,C) - g(3) - g(4)$$

$$k(s,t) = exp^{(S_{S,g}(\pi_1))} + ... + exp^{((S_{S,g}(\pi_N))}$$

Weighted Degree Position Kernel

This kernel counts the matches between two sequences s and t between the words $u_{w,i}(s)$ and $u_{w,i}(t)$ where

$u_{w,i}(s) = s_i s_{i+1} \ldots s_{i+w-1}$ for all $1 \leqslant w \leqslant d$.

The parameter w denotes the order (length of the word) to be compared [43].

The weighted degree kernel is defined as

$$k(s,t) = \sum_{w=1}^{d} w_w \sum_{i=1}^{N-d} I\left(u_{w,i}(s) = u_{w,i}(t)\right) \tag{2.27}$$

where the weighting was chosen to be $w_k = d - w + 1$, this means, higher-order matches get lower weights. This kernel emphasizes position-dependent information and decreases the influence of higher-order matches. It is similar to the spectrum kernel, however this kernel uses position-specific information [43].

An example on a weighted degree kernel $d = 3$, can be shown in table 2.10.

Table 2.10: Two protein sequences are used in weighted degree kernel, then the number of matches in 1-mers, 2-mers, 3-mers are counted respectively.

seq-name	seq
s	AAACAAATAAGTAACTAATCTTTTAGGAAGAACG
1mers	16 match
2mers	6 match
3mers	2 match
t	TACCTAATTATGAAATTAAATTTCAGTGTGCTGA

$$k(s,t) = W_1 \times 16 + W_2 + W_3 \times 2$$

And the following example in table 2.11, describes the weighted degree position kernel:

$$k(s,t) = W_{8,3} + W_{5,-4} + W_{6,3}$$

Table 2.11: Two protein sequences are used in weighted degree position kernel.

seq-name	seq
s	AAACAAATACAGTAACTAATCTTTAGTAGCAGCGAAGAAC
t	TACAAACAAATCTAATGTTTAGCTCCCAATAGCAGCGCGA

Mismatch Kernel

Mismatch kernel [28] allows some degree of mismatching in the feature map, this allowance introduce a more sensitive and biologically realistic kernel. That is, the kernel value between two sequences s and t is large if they share many similar but not identical k-mers. For a fixed k-mer $\alpha = a_1, a_2, ..., a_k$ with each a_i a character in A, the (k, m)-pattern generated by α is the set of all k-length sequences β from A that differ from α by at most m mismatches. This set is denoted by $N_{(k,m)}(\alpha)$, the 'mismatch neighborhood' around α. Also when we see an instance of a k-mer α in the input sequence s, it contributes not only to the α-coordinate in feature space but also to all coordinates corresponding to k-mers in the mismatch neighborhood of α. We can now define our feature map into the l_k-dimensional feature space, indexed as before by the set of all possible k-mers. If α is a fixed k-mer, then $\phi_(k, m)$ on α is:

$$\phi_{(k,m)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in A^k} \tag{2.28}$$

where $\phi_\beta(\alpha) = 1$ if β belongs to $N_{(k,m)}(\alpha)$ and otherwise $\phi_\beta(\alpha) = 0$ And the feature map on an input sequence s in χ is defined as the sum of the feature vectors for the $k - mers$ in s:

$$\phi_{(k,m)}(s) = \sum_{k-mers\ \alpha\ in\ s} \phi_{(k,m)}(\alpha) \tag{2.29}$$

Table 2.11: Two protein sequences are used in weighted degree position kernel.

seq-name	seq
s	AAACAAATACAGTAACTAATCTTTAGTAGCAGCGAAGCAC
t	TACAAACAAATCTAATGTTTAGCTCCCAATAGCAGCCCCA

(2.30)

Mismatch Kernel

Mismatch kernel [28] allows some degree of mismatching in the feature map: this allowance introduce a more sensitive and biologically realistic kernel. That is, the kernel value between two sequences s and t is large if they share many similar but not identical k-mers. For a fixed k-mer $\alpha = a_1 a_2 \ldots a_k$ with each a_i a character in A, the (k, m)-pattern neighborhood of α is the set of all k-length sequences β from A that differ from α in at most m mismatches. This set is denoted by $N_{(k,m)}(\alpha)$, the mismatch feature map is indexed by the set of all possible k-mers and for a given k-mer α is defined on $\beta \in A^k$ as $\phi_\alpha(\beta) = \phi_\alpha(k, m)$ where $\phi_\alpha(\beta) = 1$ if $\beta \in N_{(k,m)}(\alpha)$, i.e. if β belongs to the (k, m)-pattern neighborhood around α. Also when we see an instance of α in the sequence, it is marked as 1 otherwise it is marked as 0 in all coordinates corresponding to β in s, it contributes not only to $\phi_\alpha(k, m)$ but also to $\phi_\beta(k, m)$ where $\beta \in N_{(k,m)}(\alpha)$.

Table 2.12: Two protein sequences are used in Mismatch kernel example.

seq	
s	AACAAATAAGTAACTAATCGTTT
t	AGCCTATCATTATGAAAGTATTT

Table 2.13: The 3-mers with maximum one mismatch are listed, then if this 3-mer exists in the sequence, it is marked as 1 otherwise it is marked as 0

3-mer	ACC(AGC)	ACA	...	TAA(TCA)	AAT(AAG)	...	TTT
No in s	1	0	...	1	1	...	1
No in t	1	1	...	1	1	...	1

We can now define our feature map indexed as before by the set of all

$$\begin{aligned} k(s,t) &= < [1\,0\,\ldots\,1\,1\,\ldots\,1], [1\,0\,\ldots\,1\,1\,\ldots\,1] > \\ &= 1\times 1 + 0\times 0 + \ldots + 1\times 1 + 1\times 1 + \ldots + 1\times 1 \end{aligned}$$

TOP Kernel

It is derived from the Tangent Vectors of Posterior log-odds (TOP) and especially designed for classification, it is similar to the well-known Fisher Kernel (Jaakkola and Haussler, 1999) [21], the main idea of the TOP kernel is to incorporate prior knowledge via a given probabilistic model. It is defined as:

$$k(s,t) = \langle f_\theta(s), f_\theta(t) \rangle) \tag{2.31}$$

Here the $\phi_(k,0)$ is the spectrum kernel feature map. The $k_{(k,m)}$ mismatch kernel

$$k_{(k,m)}(s,t) = \langle \phi_{(k,m)}(s), \phi_{(k,m)}(t) \rangle \tag{2.30}$$

we note that $k_{(k,m)}(x,y)$ will be large if the sequences s and t share many k-length subsequences differing by at most m mismatches [27] [28].
The following example describes the Mismatch kernel, mer=3, max-mismatch=1:

Table 2.12: Two protein sequences are used in Mismatch kernel example.

seq-name	seq
s	AACAAATAAGTAACTAATCGTTT
t	AGCCTATCATTATGAAAGTATTT

Table 2.13: The 3-mers with maximum one mismatch are listed, then if this mers is exist in the sequence, it is marked as 1 otherwise it is marked as 0

3-mer	ACC(AGC)	ACA	...	TAA(TCA)	AAT(AAG)	...	TTT
No in s	1	0	...	1	1	...	1
No in t	1	1	...	1	1	...	1

$$k(s,t) = < [1\ 0\ ...\ 1\ 1\ ...\ 1], [1\ 0\ ...\ 1\ 1\ ...\ 1] >$$
$$= 1 \times 1 + 0 \times 0 + ... + 1 \times 1 + 1 \times 1 + ... + 1 \times 1$$

TOP Kernel

It is derived from the Tangent Vectors of Posterior log-odds (TOP) and especially designed for classification, it is similar to the well-known Fisher Kernel (Jaakkola and Haussler, 1999) [21], the main idea of the TOP kernel is to incorporate prior knowledge via a given probabilistic model. It is defined as:

$$k(s,t) = \langle f_\theta(s), f_\theta(t) \rangle) \tag{2.31}$$

where

$$f_\theta(s) := (v(s,\theta), \partial_{\theta 1} v(s,\theta), ..., \partial_{\theta p} v(s,\theta))^{\mathrm{T}} \tag{2.32}$$

and

$$v(s,\theta) = \log\left(P(y=+1|s,\theta)\right) - \log(P(y=-1|s,\theta)) \tag{2.33}$$

while

$$\partial_{i,j} v(s,\theta) = I(si=j)/\theta_{i,j} \tag{2.34}$$

and hence the kernel is computed as:

$$k(s,t) = v(s,\theta)v(t,\theta) \tag{2.35}$$

Salzberg Kernel

In 1997, Salzberg developed positional conditional probability matrix that takes into account the dependency between adjacent bases [46], however Zien *et al.* introduces a modification for Salzberg method that brings the *"Salzberg kernel"*, So instead of calculating the product of the conditional probabilities over the whole sequence, calculate the log odds of the conditional probabilities for each position separately. Salzberg kernel is similar to the locality-improved kernel. while it is applied on the sequences of log odds scores $s_p(x)$, which is defined as:

$$s_p(x) = \log \frac{P(x_p \; at \; pos. \; p \; in \; Truedata | x_{p-1} at \; pos. \; p-1 \; in \; Truedata)}{P(x_p at \; pos. \; p \; in \; Alldata | x_{p-1} \; at \; pos. \; p-1 \; in \; Alldata)} \tag{2.36}$$

Where, P is the estimated probabilities derived from training set counts plus pseudo counts, x_p is the amino acid incident at position p in the sequence corresponding to data point x, Truedata is the set of training sequences

belongs to positive samples, and Alldata is the set of all training sequences (both positive or negative). So here a new input space is defined as each data point is represented by a sequence of log odd scores $s_p(x)$ relating, individually for each position, two probabilities: first, how likely the observed amino acid at that position derives from the positive data and second, how likely that amino acid occurs at the given position relative to all set of training sequences [65].

2.2 Support Vector Machine (SVM)

SVM is a supervised learning technique that generates input-output mapping relations from a set of labeled training data. It is an example of a linear classifiers that is the only one that maximizes the margin (maximizes the distance between it and the nearest data point of each class). This linear classifier is termed the optimal separating hyperplane that generalize well as opposed to the other possible classifiers.

SVM can be either applied to classification problems or regression problems [14].In classification, nonlinear kernel functions are often used to transform data to a high dimensional feature space, in which the input data become more separable, compared to the original input space. Intuitively, SVM with a maximum margin is then created. SVM is based on structural risk minimization principle from statistical learning theory [4, 14]

The data for a two-class learning problem consists of objects labeled with one of two labels; for convenience we assume the labels are +1 (positive examples) and -1 (negative examples). Then, the classification steps of these two sets involve mapping them using kernel function, onto high dimensional space. Then finding the optimal hyperplane that differentiate the two classes

with maximal margin.

Let x be a point in an M-dimensional vector space, it denotes a vector with M components x_j, $j = 1, ..., M$. The notation x_i will denote the i^{th} vector in a dataset $\{(x_i, y_i)\}_{i=1}^{n}$, where y_i is the label associated with x_i, and n is the number of examples. The objects x_i are called patterns, inputs, and also examples. In general linear classifier can be defined as the dot product between two vectors:

$$\langle w, x \rangle = \sum_{j=1}^{M} w_j x_j \tag{2.37}$$

A linear classifier is based on a linear discriminant function of the form

$$f(x) = \langle w, x \rangle + b \tag{2.38}$$

The function $f(x)$ assigns scores for a point x, then classifies the point according to this score, the vector w is the weight vector, and the scalar b is the bias that translates the hyperplane with respect to the origin. The points satisfying the equation $\langle w, x \rangle = 0$ correspond to a line through the origin in the two dimension , and a hyperplane in three dimension.

The hyperplane divides the space into two half spaces according to the sign of $f(x)$, that indicates on which side of the hyperplane a point is located, if $f(x) > 0$, then one decides for the positive class, otherwise for the negative. The boundary between regions classified as positive and negative is called the decision boundary of the classifier.

In case of the linear separable data, there exists many of the hyperplanes that correctly classifies these data points, however we should choose the hyperplane that grantee classifying the unseen examples correctly. The optimal hyperplane classifier not only separates the examples correctly, but does so with a large margin, as suggested by the statistical learning theory [58]. Fig-

ure 2.3 illustrates the maximum margin classifier (SVM).

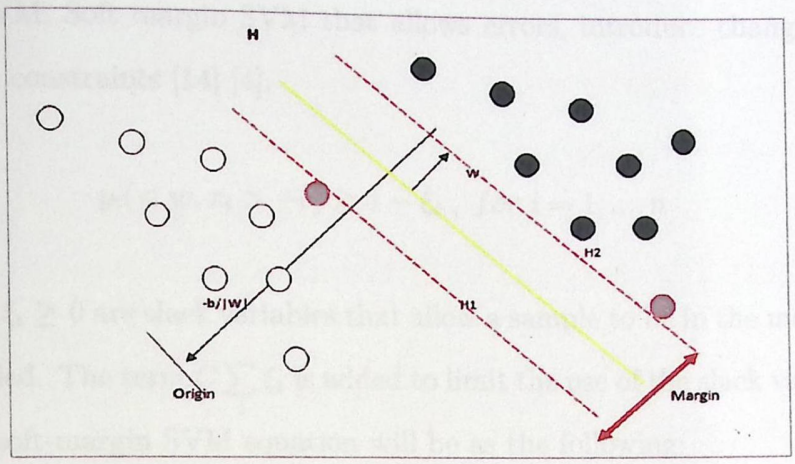


Figure 2.3: The maximum margin boundary generated by a linear SVM.

The margin of a linear classifier is defined as the distance of the closest example to the decision boundary margin is $\frac{1}{\|w\|}$, where $\|w\|$ is the length of w, also known as its norm, given by $\sqrt{\langle w, w \rangle}$.

The classifier that is called hard margin SVM, applicable to linearly separable data, is the classifier with maximum margin that correctly classify all the input examples. To find the optimal w and b corresponding to the maximum margin hyperplane, one has to solve the following optimization problem:

$$\begin{aligned} &\underset{w,b}{\text{minimize}}\ \tfrac{1}{2}\|w\|^2 \\ &\textit{subject to}: y_i(< w, x_i > +b) \geq 1, \\ &\forall\, i = 1, ..., n \end{aligned} \tag{2.39}$$

Minimizing $\|w\|^2$ is equivalent to maximizing the margin, where the constraints ensure that each example is correctly classified. This type of problems called a quadratic optimization problem, in which the optimal solution (w, b) is found satisfying the above mentioned constraints.

In practice, data are often not linearly separable; and even if they are, SVM provide a greater margin that allows the classifier to misclassify some

points, however it will generally provide better performance than the hard margin SVM. Soft margin SVM that allows errors, introduce changing the inequality constraints [14] [4].

$$y_i(< w, x_i > +b) \geq 1 - \xi_i \ , \ for\ i = 1, ..., n \tag{2.40}$$

where $\xi_i \geq 0$ are slack variables that allow a sample to be in the margin or misclassified. The term $C \sum_i \xi_i$ is added to limit the use of the slack variables, then the soft-margin SVM equation will be as the following:

$$\begin{aligned} &\underset{w,b}{\text{minimize}}\ \tfrac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\ &subject\ to\ : y_i(< w, x_i > +b) \geq 1 - \xi_i, \\ &\xi_i \geq 0\ for\ i = 1, ..., n \end{aligned} \tag{2.41}$$

The constant $C > 0$ sets the relative importance of maximizing the margin and minimizing the amount of slack. In order to solve this optimization problem, the method of Lagrange multipliers is used, it reformulates the original primary problem into dual formalization, it is expressed in terms of α_i as in the following equation:

$$\begin{aligned} &\underset{\alpha}{\text{maximize}} \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_j, x_j \rangle \\ &subject\ to : \sum_{i=1}^{n} y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C. \end{aligned} \tag{2.42}$$

then the weight vector w can be expressed using the samples x_i and the solution α_i as the following:

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i \tag{2.43}$$

All the samples x_i which posses $\alpha_i > 0$ are called support vectors, they

are the points which lie on the margin boundaries. Intuitively, the support vectors do contribute in formulating the geometric location of the large margin hyperplane, and whose removal affects greatly the solution. SVM can be applicable to non linearly separable data, by mapping the data to some vector space using mapping function ϕ, furthermore, SVM can handle domains such as biological sequences where a vector space representation is not necessarily available [4].

The discriminant function will be as the following:

$$f(x) = \langle w, \phi(x) \rangle + b \tag{2.44}$$

Note that $f(x)$ is a linear function in the new feature space defined by the mapping ϕ, while it is nonlinear in the original space if $\phi(x)$ is a nonlinear function. The mapping can be done using kernels, as it is shown in the following equations when we reformulate the equation of the weighting vector w, then substituting it in the discriminant function.

$$w = \sum_{i=1}^{n} y_i \alpha_i \phi(x_i) \tag{2.45}$$

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i \langle \phi(x_i), \phi(x) \rangle + b \tag{2.46}$$

As we know that the kernel function $k(x, x_i)$ defined as

$$k(x, x_i) = \langle \phi(x), \phi(x_i) \rangle \tag{2.47}$$

then the dual formulation for the discriminant function becomes useful, as it solves the problem using the kernel function which presents a meaningful similarity measures and can be computed efficiently [4].

2.3 Evolutionary Approaches

Evolutionary Algorithms (EA) are stochastic optimization techniques based on the principles of natural evolution [7].

The EA maintains a collection of potential solutions to a problem. Some of these possible solutions are used to create new potential solutions through the use of operators such as crossover and mutation. Operators act on and produce collections of potential solutions. These potential solutions are selected on the basis of their quality as solutions to the problem using fitness function. The fitness function is a measure of how good the solution is, and how far it is from the optimized solution. EA uses this process iteratively to generate new collections of potential solutions until some stopping criterion is met [7].

The classical families of evolutionary algorithms are: Evolutionary Programming (EP), Genetic Programming (GP) [24], Evolution Strategies (ES) [13], Genetic Algorithms (GA), Evolution Programs, and Memetic Algorithms (MA) [7]. In addition to the different EA variants mentioned above, there exist several other techniques that could also fall within the scope of EAs, such as Ant Colony Optimization and Particle Swarm Optimization [53]. In our approach we used both the Genetic Programming technique (GP) and the Genetic Algorithm technique (GA) to find the optimized kernel by combining several string kernels.

2.3.1 Genetic Algorithm

GA is a technique that inspired by the evolution of the biological behavior, it is efficient and easy technique that gives the best performance in the domain of optimization and searching. However, GA is a stochastic method

and generally a time consuming method. The evolution of GA starts by an initial random population. Generations are simulated by mating the best individuals in the populations and applying some biological operations on the resulted offspring. Mutation and Crossover are two famous operations that help us walk through the search space looking for the global optimal value. The fittest individuals are adopted to constitute the next generation. Algorithm 1 demonstrates the procedure of GA [34].

Algorithm 1 GA basic steps

```
begin
t ← 0
initialize P(t)
evaluate P(t)
while (not termination condition) do
  begin
  t ← t + 1
  select P(t) from P(t-1)
  alter P(t)
  evaluate P(t)
end while
end
```

Fitness Function

The fitness function is the most important concept in GA. It is a measure of how much the individual is far from the best solution by giving each individual a quantitative weight, this weight determines how well an individual is able to solve the problem. The fitness function is a problem oriented aspect, and it minimizes or maximizes the search toward the fittest solution.

Individual Encoding

GA individuals can be encoded in an suitable way that enables the GA to behave positively during its execution. There are many formats of encoding,

the most used one is the binary encoding. Other formats of encodings represents the individual by an array of integers or decimal numbers, and others encodes the individual to a string of letters.

Selection

There are many strategies for selecting individuals for survival and reproduction. These methods include:

- Proportional Selection: a famous method by which the selection is done randomly, and fittest chromosomes have higher probabilities for being selected. This method is also called "Roulette Wheel" [34].
- Tournament selection: some number of individuals usually used to compete for selection to the next generation, this competition(tournament) is repeated number of times equals to population size [34].
- Ranking Methods: all individuals in the populations are sorted from the best to the worst and probabilities if their selection are fixed for the whole evolution process [34].

Genetic Operators

There are two methods that take place in the evolution process to simulate the biological nature; Crossover and Mutation. In crossover two individuals mate by swapping segments of their codes to produce an artificial offspring. The idea behind crossover is that the new individual may be better than either parent if it takes the best characteristics from each of them. Crossover occurs during evolution according to a user-defined probability. Three strategies of crossover including single-point, two-point, and uniform crossover.

the most used one is the binary encoding. Other formats of encodings represents the individual by an array of integers or decimal numbers, and others encodes the individual to a string of letters.

Selection

There are many strategies for selecting individuals for survival and reproduction. These methods include:

- Proportional Selection: a famous method by which the selection is done randomly, and fittest chromosomes have higher probabilities for being selected. This method is also called "Roulette Wheel" [34].
- Tournament selection: some number of individuals usually used to compete for selection to the next generation, this competition(tournament) is repeated number of times equals to population size [34].
- Ranking Methods: all individuals in the populations are sorted from the best to the worst and probabilities if their selection are fixed for the whole evolution process [34].

Genetic Operators

There are two methods that take place in the evolution process to simulate the biological nature; Crossover and Mutation. In crossover two individuals mate by swapping segments of their codes to produce an artificial offspring. The idea behind crossover is that the new individual may be better than either parent if it takes the best characteristics from each of them. Crossover occurs during evolution according to a user-defined probability. Three strategies of crossover including single-point, two-point, and uniform crossover.

After a crossover is performed, mutation take place. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding a switching few randomly chosen bits from 1 to 0 or from 0 to 1.

Population Size

A population is the set of individuals in one generation. Population size represents the number of individuals in a population. The larger the population size, the better the chance that an optimal solution will be found.

Population diversity

Diversity refers to the average distance between individuals in the population, it is one of the most important factors that determine the performance of the GA. It enables the algorithm to search a larger region of the search space.

Stopping Criteria

The genetic algorithm stops when it reaches one of the following conditions: number of generations, time limit, fitness limit, and stall time limit.

2.3.2 Genetic Programming GP

A genetic programming algorithm GP [24] is a variant of evolutionary optimization algorithms. It is a special case of GA. While GA works with binary strings, GP use trees as structure for individuals and it represents the solution as a hierarchical computer programs with dynamical varying sizes and shapes. GP applies the approach of the genetic algorithm to the space of possible computer programs A wide variety of seemingly different problems

After a crossover is performed, mutation take place. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding a switching few randomly chosen bits from 1 to 0 or from 0 to 1.

Population Size

A population is the set of individuals in one generation. Population size represents the number of individuals in a population. The larger the population size, the better the chance that an optimal solution will be found.

Population diversity

Diversity refers to the average distance between individuals in the population, it is one of the most important factors that determine the performance of the GA. It enables the algorithm to search a larger region of the search space.

Stopping Criteria

The genetic algorithm stops when it reaches one of the following conditions: number of generations, time limit, fitness limit, and stall time limit.

2.3.2 Genetic Programming GP

A genetic programming algorithm GP [24] is a variant of evolutionary optimization algorithms. It is a special case of GA. While GA works with binary strings, GP use trees as structure for individuals and it represents the solution as a hierarchical computer programs with dynamical varying sizes and shapes. GP applies the approach of the genetic algorithm to the space of possible computer programs A wide variety of seemingly different problems

from many different fields can be reformulated as a search for a computer program to solve the problem.

A simple description of a GP algorithm is given in Algorithm 2 .

Algorithm 2 GP basic algorithm

```
Initialize population with random candidate solutions
Evaluate each candidate using fitness function
while (not termination criteria is reached) do
  begin
  Select parents
  Recombine pairs of parents
  Apply genetic operation on the resulting offspring (crossover, mutation,
  selection)
  Evaluate new candidates using the fitness
  Select individuals for the next generation
end while
```

A randomly generated population is used to initialize the evolutionary process. Then each candidate solution is evaluated by a fitness function in each iteration (generation) of the algorithm, the candidate solutions with the highest fitness have a higher probability of being selected in the next generation.

A genetic operations are then applied on the candidate solutions in every generation which move the search forward. The operations could be crossover, mutation and selection. In crossover, a new solutions are created by combining parts of two or more solutions. However in mutation, the solution is randomly altered see Figure 2.4 , Figure 2.5. In the selection, the candidate solutions that will be the basis for the next generation are being selected. Genetic operators need parent individuals to produce their children. These parents are selected according to one of four sampling methods:

- Roulette: Each individual owns a portion of the roulette that corresponds to its expected number of children then the roulette with random pointers is spun.

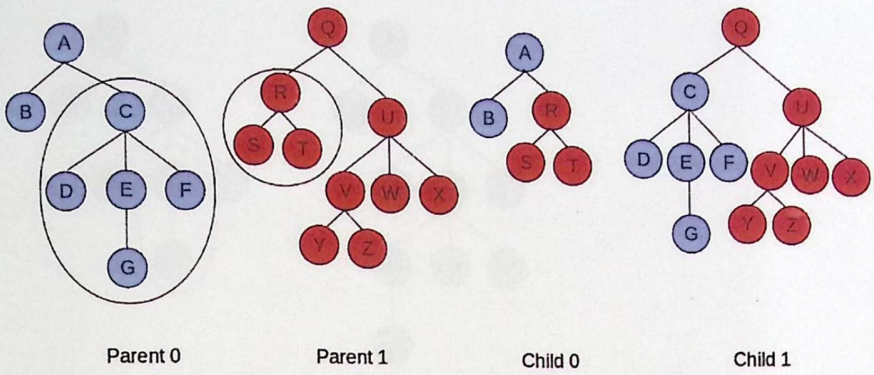


Figure 2.4: Two individuals mate by swapping segments of their codes to produce an artificial offspring.

- Sus: It relies on the roulette, but the pointers are equally spaced.
- Tournament: Each parent is chosen by randomly drawing a number of individuals from the population and selecting only the best of them.
- Lexicographic parsimony pressure is implemented in this method. a random number of individuals are chosen from the population and the best of them is chosen, however if two individuals are equally fit, the shortest one (the tree with less nodes) is chosen as the best.

2.4 Classification Performance Measures

The Performance measures are statistical techniques used to evaluate a classifier. There is a large variation in the measures used to asses prediction systems in machine learning.

Here we introduce some description for the measures used in our experiments,

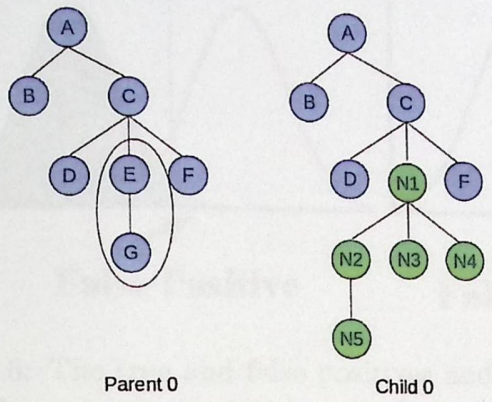


Figure 2.5: Mutation changes randomly the new offspring, the solution is randomly altered

that we rely on to evaluate the classifier performance along with our evolved kernel.

Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a true positive (TP); if it is classified as negative, it is counted as a false negative (FN). If the instance is negative and it is classified as negative, it is counted as a true negative (TN); if it is classified as positive, it is counted as a false positive (FP). See Figure 2.6.

These metrics forms the basis for many common measures as follows [11]. The true positive rate (also called hit rate and Recall) of a classifier is:

$$\text{Tp} - \text{Rate} = \frac{Positives\ correctly\ classified}{Total\ positives} \tag{2.48}$$

The Positive Predictive Value (also called Precision) can be calculated as

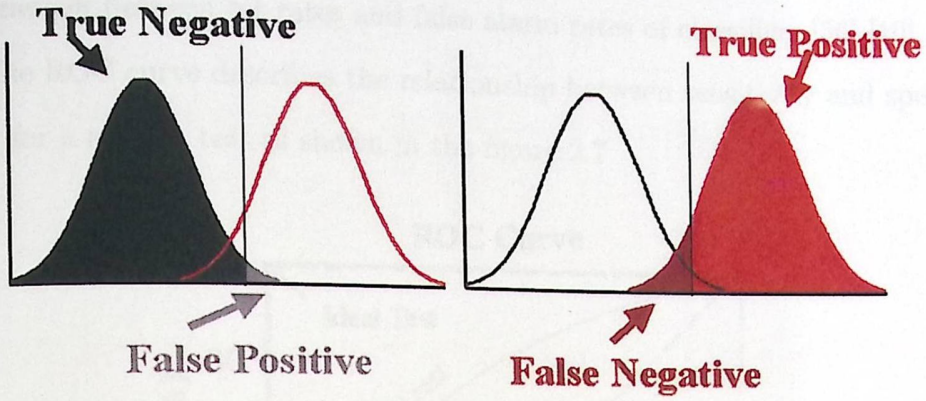


Figure 2.6: The true and false positives and negatives is illlustrated in this figure, when a predictive test is a continuous measurement, the true and false positives and negatives depends on the cut off point for test [39]

follows:

$$\text{Precision} = \frac{True\ Positives}{True\ Positives\ +\ FalsePositives} \tag{2.49}$$

The false positive rate (also called false alarm rate) of the classifier is

$$\text{FP} - \text{Rate} = \frac{Negatives\ incorrectly\ classified}{Total\ negatives} \tag{2.50}$$

Also the sensitivity and specificity are associated with ROC curves,

$$Sensitivity = Recall \tag{2.51}$$

$$\text{Specificity} = \frac{TrueNegatives}{FalsePositives\ +\ TrueNegatives} \tag{2.52}$$

The most commonly used technique for evaluating classifiers and visualizing their performance, are the Receiver Operating Characteristics (ROC) graphs.

ROC graphs have long been used in signal detection theory to describe

the tradeoff between hit rates and false alarm rates of classifiers [56] [10].

The ROC curve describes the relationship between sensitivity and specificity for a specific test as shown in the figure 2.7.

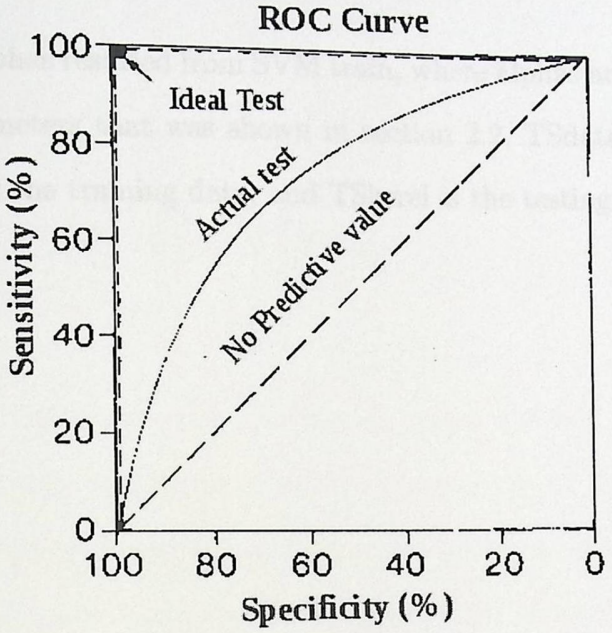


Figure 2.7: The ideal test produces 100% sensitivity and 100% specificity as shown. If a the test has no predictive value, and the result is obtained by a random selection process, the relationship between sensitivity and specificity is linear as shown. The observer determines the actual operating point along this line [39].

This graph shows a comparison of ROC curves for an ideal test procedure with one that produces no useful information. The ideal test produces 100% sensitivity and 100% specificity. If a test procedure has no predictive value, and the diagnosis is obtained by a random selection process, the relationship between sensitivity and specificity is linear.

In SVM, the score values that are needed to plot the ROC curve are representing the distance from any data point to the hyperplane, this score

values is calculated in our experiments accordig to the following equation:

$$\begin{aligned} Score = {} & bias \times ones(1, length(TSdata)) + alphas(:,1)' \\ & \times TSkernel(TRdata(1 + alphas(:,2)'), TSdata) \end{aligned} \tag{2.53}$$

The bias and alphas resulted from SVM train, where alphas are the Lagrange multiplier parameters that was shown in section 2.2, TSdata is the testing data, TRdata is the training data, and TSkerel is the testing kernel.

Chapter 3

Literature Review

The problem of optimizing multiple kernels, takes different point of views, some research make optimization for the parameters of a specific kernel, and others combine different numerical kernels using one of the evolutionary approaches or other optimization techniques. This chapter introduces the earlier work related to combining evolutionary approaches with the kernels methods with the purpose of kernel evolution and optimization. The first section gives a view on the previous work that evolves numerical kernel, section 3.2 introduces a previous work related to optimizing kernel parameters using evolutionary approaches. Section 3.3 talks about previous work on protein prediction. Section 3.4 demonstrates our contribution in this thesis, showing the novelty in evolving a new string kernel and optimizing its parameters.

3.1 Numerical kernel evolution

There is continuous development in the field of kernel optimization using the different evolutionary approaches. Though most methods are still make optimization for numerical kernels, by finding the best combination of these kernels.

Methastate et al. [32] introduced a family-based GP for generating the optimal kernel. They used the ES to investigate the optimal parameter space. The evolved kernel is built using the two types of operators: addition and multiplication, while the kernel function is one of three types of numerical functions: linear, RBF and polynomial function. Their tree family denotes for a group of structurally identical trees, while their kernel tree composed of a multi-kernel function in the form of tree structure. They introduced a normalized structure of the kernel tree to preserve the structural regularity, by reordering its branches according to their weights.

Sullivan et al. [54] introduced a new algorithm called KGP that used Strongly Typed Genetic Programming (STGP) and principled kernel closure properties to find near optimal kernels. They used the basic kernel functions as a terminal set, which are Polynomial, Gaussian and Sigmoid with randomly-chosen parameters. Lexicographic tournament selection was used in their approach to prevent growth of the kernel functions without any increase in fitness, however STGP ensures that selection, crossover, and mutation will not generate an invalid kernel function. The fitness function is the average of k-fold cross validation on the experimented data set.

Diosan et al. [9] designed a Complex Multiple Kernel (CMK) and optimized its parameters using GP and apply it to SVM. The chromosome is a tree that encodes the mathematical expression of MK, It is a combination simple numerical kernels (linear, polynomial, RBF, Sigmoid) and some offset shifting coefficients, with additional parameters added to each kernel, using addition, multiplication, and exponential mathematical operators. Their hybrid model discovered the optimal expression of MK function and optimized

its parameters, also tuned the regularization parameter C. They called the optimal found kernel a (eCMK). They used the classification accuracy as a fitness function that evaluated each individual using SVM.

Howley et al. [18] introduced a technique called genetic kernel SVM (GKSVM), where they used the genetic programming to evolve kernel from basic kernels polynomial, RBF and sigmoid kernels. They create a kernel trees composed of kernel functions, their fitness function is the training error, however when two trees posses the same error another fitness was used based on the sum of support vectors. Then they presented a KTree model [19] that used genetic programming for constructing the kernels as a modified and extended version of GKSVM [18]. Their new model used a more sophisticated kernel representation that can represent standard kernels, such as RBF, used a Mercer filter to improve performance, and also it used a different fitness functions based on cross validation. They divided the kernel tree into two parts: vector, scalar trees, the input to the vector tree are the two samples, these inputs were combined using an addition or subtraction operations, while exponential and tanh operations can be used in the scalar tree. Then they applied a mercer filter on this tree by deleting the kernel tree that contained any negative eigenvalue . Three fitness functions were used , one of them was the training set classification error in combination with a different tiebreaker fitness, such as the kernel tree size, sum of the support vector values,and a 3-fold cross-validation test on the training data along with tree size.

Phienthrakul et al. [41] proposed a new classification model called GPES that combines GP and ES to evolve hybrid kernel which satisfies Mercers theorem and can be used used in SVM, their hybrid kernel expressed as a

tree since it has some adjustable parameters. GP used for creating the hybrid kernel and ES used for adjusting the sub-kernel weights and parameters, the sub-kernel could be Polynomial or RBF. The operations used in the tree are "plus" and "multiplication". They used the bound of generalization error as a fitness function for evaluating their hybrid kernel, which indicates the capacity of the machine to classify the data.

Friedrichs et al. [12] used the covariance matrix adaptation evolution strategy (CMA-ES) to select the kernel from a parameterized kernel space and to control the multiple SVM hyperparameters. They applied their method on Gaussian kernels.

Methasate et al. [33] proposed a kernel-tree method whose function composed of multiple kernels in the form of tree structure. They used GP to find the optimal tree structure and its parameters. They also used gradient decent method to tune the kernel parameters. The operation node can be either of the addition or multiplication operators on the basic functions: Linear, Polynomial, and RBF kernels. They combine the advantages of GP and gradient search to find the global optimal solution with its optimal parameters.

As an example for researches that creates new kernels using approaches other than evolutionary, we will mention the work of K.Crammer et al. [8] as they constructed an accurate kernel from weighted combination of kernels using boosting framework. They introduce a new algorithmic implementation of the base learning algorithm for kernels, that had an influence in returning a good kernel from the boosting algorithm.

Another development in the field of kernels is the use of multiple kernels instead of a single one, that is known by multiple kernel learning problem (MKL). MKL can improve classifier performance, as it was introduced by Lanckriet et al. [25], they used semi-definite programming in solving it. MKL problem was reformulated in many researches, one of them based on a semi-infinite linear problem (SILP) that has been proposed by Sonnenburg et al [52], their algorithm solves the problem by iteratively solving a classical SVM problem with a single kernel and a linear programming for which number of constraints increases along with iterations.

3.2 Kernel parameter optimization

An optimization for the parameters of a kernel function is presented by many researchers using different approaches. Many of them used the evolutionary approaches, others used gradient decent approaches or the combination of them. Regarding these researches Mersch et al. [31] introduced a new parameterization for a k-mer oligo string kernel, where all oligomers of length k are weighted individually, based on evolution strategy. They used the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), an adaptive variable metric algorithm for efficient direct real valued optimization to search for appropriate hyperparameters. The different k-mers are given a new weight in the oligo kernel because they may be more discriminative than others, then evolve the combined weighted oligo kernel.

Ahn et al. [1] introduced a model called (SOSVM) Simultaneous Optimization of SVM using GA, they optimized the feature selection and the

instance selection, as well as the parameters of the kernel function used in SVM, using GA. They used the Gaussian RBF as the kernel function, and tuned its parameter δ^2 and the SVM parameter C. Their fitness function is the prediction accuracy for the testing dataset.

Chapelle et al. [6] introduced a system that automatically tunes multiple parameters for SVMs and kernel, their model used the idea of minimizing some estimates of the generalization error of SVMs using a gradient descent algorithm that improved the performance and reduced the complexity of the solution. They perform an experiment in many applications to adjust the parameter C in SVM, and the parameter σ for RBF kernel.

Phienthrakul et al. [40] create a model that produces a flexible kernel function by combining multi-scale Radial Basis Function (RBF) kernels linearly, including weights. ES are used to adjust sub-kernel weights and the widths of the RBF kernels.

Other researches that one could pay attention to, are those who build a combined models of evolutionary systems and machine learning systems such as (GA/SVM) hybrid model that is proposed by Zhao et al. [64]. Their system was used to classify proteins by selecting features from the protein sequences and then used it to train SVM classifier simultaneously using GA. They found the optimal feature vector among set of feature vectors extracted from protein sequences, then they used this optimal vector to train SVM in order to classify newly protein sequences. Their fitness was a multi objective function that minimized the feature numbers and maximized the classification accuracy.

3.3 Protein sequences prediction

Machine learning approaches have been extensively used in developing methods for signal peptide prediction. Signal peptide have an immense impact on modern cell biology as they are responsible of the access of many proteins to the secretory pathway. Many researchers have been interested in building models that predicts signal peptide and their cleavage cites using the different machine learning approaches, some of these methods are based on Hidden Markov Models (HMM) as the models introduced by Kall et al. [23], they propose a predictor is based on HMM that models the different sequence regions of a signal peptide and the different regions of a transmembrane protein in a series of interconnected states. their model is a combined transmembrane protein topology and signal peptide predictor. In the same field, Zou et al. [66] presented a HMM that combined a transmembrane barrel submodel and signal peptide submodel for both topology and signal peptide prediction. Signal peptide predictors were also handled using neural network, as it was introduced by Nielsën et al. [38] [37], they developed a system for predicting signal peptides and their cleavage sites, called SignalP. A combined feed-forward neural network approach had been presented to recognize signal peptides and their cleavage sites, using one network to recognize the cleavage site and another network to distinguish between signal peptides and non-signal peptides. Also they developed a HMM version of SignalP in [36], this model able to discriminate between cleaved signal peptides and uncleaved signal anchors. Another neural network based method was developed by Plewczynski et al. [42], their method used two neural net-

works trained on experimental data from Swisprot database. Recently, SVM was used intensively to solve the signal peptide prediction problem.

Mukherjee et al. [35] used SVM to predict secretory signal peptides for both prokaryotic and eukaryotic signal organisms, also they predict their cleavage sites. They use two types of kernels one based upon Hamming distances and one based upon a similarity matrix, the Percent Accepted Mutation (PAM) Matrix. The PAM matrix can be thought of as the probability that one amino acid replaces another so the similarity between two amino acids acids. Sun st al. [55] use the SVM to predict signal peptide and its cleavage sites, they divided the protein sequence into two segments and then calculated the amino acid compositions on both segments, then they formulated the feature vectors from the pseudo amino acid compositions (PseAAs). Vert et al. [60] introduced a new class of string kernels derived from either from either independent probabilities or from Markov models probabilistic models. Then a SVM was used with the new kernel to predict signal peptides and their cleavage sites.

Wang et al [62] introduced a string kernels for protein sequence based on the subsite coupling model. the constructed kernel was used with SVM to predict the cleavage site of signal peptides from the protein sequences. They introduced a generalized probability kernel, and they defined their mapping function as a probabilistic model based on coupling among three key subsites.

3.4 Thesis Contribution

As we see from the above previous work and up to our knowledge, no attempts had been done to evolve a string kernel. We introduce an evolutionary based approach that can generate and optimize novel string kernels, using two evo-

lutionary approaches, the GA approach, and the GP approach. This model can optimize sub-kernel parameters, and SVM regularization parameter, it can be then used to classify biological sequences.

Our kernel tree chromosome encodes the mathematical expression of combined string kernels. We evolve more than one form of the new string kernel, some of them is a weighted sum of sub-kernels, others can be seen as mathematical expression composed of string kernels combined with addition, multiplication, exponential operations.

Each tree is evaluated using a good fitness function, that does not rely on the classification accuracy that causes overfitting. However, it considers both the sensitivity and specificity in evaluating the kernel tree. Suitable sampling methods is used in our model thats control bloat in the tree, also suitable crossover and mutation probability is used to help in producing the best kernel combination.

Our system selects its terminal set from the most famous string kernels which each of them has set of parameters that need strongly to be optimized, and also selected the mathematical operation from set of the valid kernel operations that preserves Mercer's theorem.

We use our new model in classifying biological sequences, especially in predicting signal peptide in mammalian proteins to determine whether they are is a secreted proteins or not, and in predicting MHCII peptides to distinguish between binding and non-binding peptides.

Chapter 4

Evolutionary Approaches for Kernel Optimization

Our main contribution is to evolve a new string kernel by combining a set of string kernels and to optimize both the sub-kernel hyperparameters and the SVM regularization parameter using two evolutionary approaches, the Genetic Programming and Genetic Algorithms. We have tested our new evolved kernel on predicting signal peptide in proteins to determine whether it is a secreted protein or not. Figure 4.1 shows a general block diagram of the proposed system. This chapter explains the two approaches taken for all the experiments and shows how the results are generated and validated.

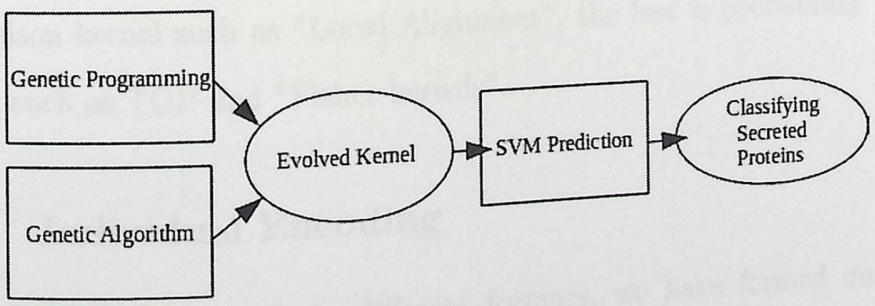


Figure 4.1: General Block Diagram of our System

4.1 Genetic Programming Approach

Our proposed methodology uses one of the most powerful evolutionary approaches; the GP. It is used for evolving the new string kernel and optimizing it parameters , then it is used along with optimized SVM to predict mammalian secreted proteins.

In GP, a potential solution is a tree encoding the mathematical representation of the evolved kernel and its parameters. The tree-based representation in GP for our based evolved kernel, introduces larger search space than other representation like arrays. The leaves of our trees are selected from a set of predefined string kernels, where the predefined string kernel is a function with two arguments represents two sequences.

The sub-kernel was chosen from a set of kernels called Terminal Set TS. While the tree nodes contain elements from a set of functions called FS, contains the mathematical operations that preserves Mercer theorem and Closure properties [43]. The TS contains sub-kernels that have been chosen from the most famous string kernel in the bioinformatics field. Our string kernels covers all types of string kernels mentioned in section 2.1.7, as string kernels in general fall into four main categories,the first is the substring kernels such as "Spectrum" and "Mismatch", the second category is the position dependence kernels such as "Weighted Degree" , the third category is the pairwise comparison kernel such as "Local Alignment", the last is probability based kernels such as TOP and "Fisher kernels".

4.1.1 Individual Encoding

The GP tree can be built in different formats, we have formed our tree either from a weighted sum combination of kernels, or from a mathematical

4.1 Genetic Programming Approach

Our proposed methodology uses one of the most powerful evolutionary approaches; the GP. It is used for evolving the new string kernel and optimizing it parameters , then it is used along with optimized SVM to predict mammalian secreted proteins.

In GP, a potential solution is a tree encoding the mathematical representation of the evolved kernel and its parameters. The tree-based representation in GP for our based evolved kernel, introduces larger search space than other representation like arrays. The leaves of our trees are selected from a set of predefined string kernels, where the predefined string kernel is a function with two arguments represents two sequences.

The sub-kernel was chosen from a set of kernels called Terminal Set TS. While the tree nodes contain elements from a set of functions called FS, contains the mathematical operations that preserves Mercer theorem and Closure properties [43]. The TS contains sub-kernels that have been chosen from the most famous string kernel in the bioinformatics field. Our string kernels covers all types of string kernels mentioned in section 2.1.7, as string kernels in general fall into four main categories,the first is the substring kernels such as "Spectrum" and "Mismatch", the second category is the position dependence kernels such as "Weighted Degree" , the third category is the pairwise comparison kernel such as "Local Alignment", the last is probability based kernels such as TOP and "Fisher kernels".

4.1.1 Individual Encoding

The GP tree can be built in different formats, we have formed our tree either from a weighted sum combination of kernels, or from a mathematical

expression including the plus, multiplication, exponential operations. So the two forms of trees will be introduced in the following subsections.

Evolve a string kernel as a weighted-sum combination

The individual here was represented as a weighted sum combination of string kernels, as we used only summation operation, and we gave a weight parameter for each kernel used. This weight parameter indicates the role of each kernel in producing the evolved kernel. An example of a GP potential solution is:

$$a_1 k_1(x_1...x_h) + a_2 k_2(x_2...x_u) + ... + a_n k_n(x_n...x_m) \tag{4.1}$$

where a_i are weights, which are adjusted during the evolution of the solution. k_i are different kernel types and $x_i...x_j$ are their corresponding parameters which are also selected by evolution. Figure 4.2 shows a sketch for the weighted sum tree example.

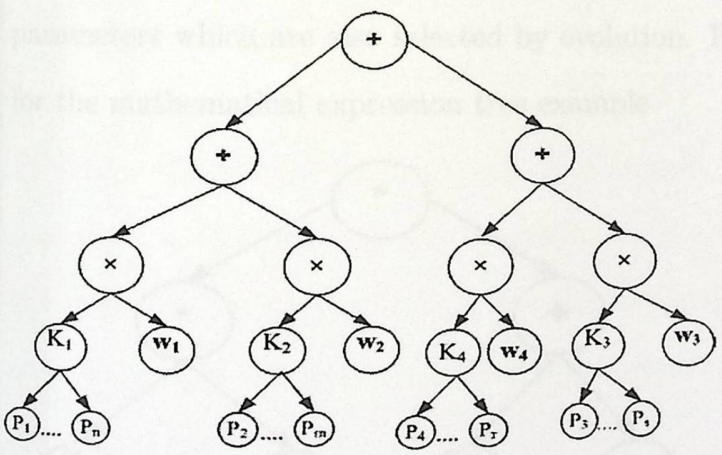


Figure 4.2: An individual that represents a kernel tree composed of weighted sum of string kernels

GP generates new string kernel by searching for the best combination of predefined string kernels. GP first initializes its population with different combination of predefined kernels with weight parameters, then it makes its

selection according to the fitness function, which is in our case the accuracy rate resulted from SVM.

GP then performs a crossover and mutation. This process is repeated until an optimal kernel is found.

Evolve a string kernel as mathematical expression

The individual that represents our evolved kernel can take another form, it was formulated as mathematical expression that contain addition, multiplication, and exponential operations. These operations preserves closure properties that verifies that the resulted kernel is a Mercer kernel. An example of a GP potential solution is:

$$k_1(x_1...x_h) \times \exp(k_2(x_2...x_u) + k_3(x_3...x_y)... \times k_n(x_n...x_m) \qquad (4.2)$$

where k_i are different kernel types and $x_i...x_j$ are their corresponding parameters which are also selected by evolution. Figure 4.3 shows a sketch for the mathematical expression tree example.

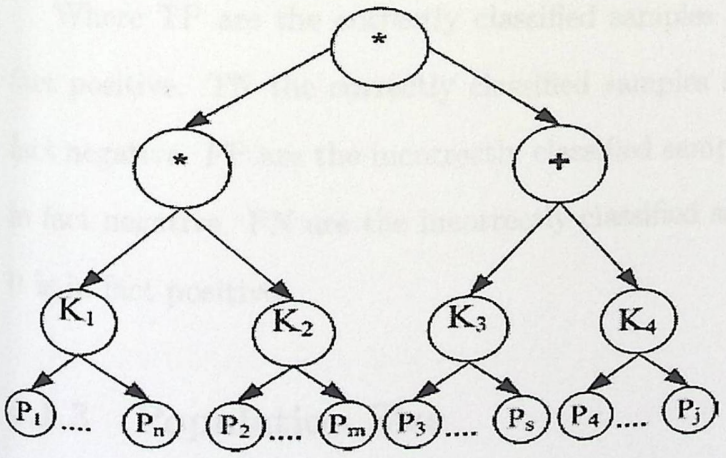


Figure 4.3: An individual that represents a kernel tree as mathematical expression of string kernels

4.1.2 Fitness Function

Our fitness function is based on the validation process of the solution. We used one fold cross validation technique, a random 1/3 of the data was used for testing and the rest of data used for training SVM with the resulting string kernel. The individual has been processed before the evaluation process. Algorithm 3 shows the individual processing and Algorithm 4 demonstrates the fitness procedure that was used to evaluate the individual. The fitness of the solution can be measured by the multiplying the sensitivity by the specificity according to the following equation:

$$Fitness = Sensitivity \times Specificity \tag{4.3}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{4.4}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{4.5}$$

Where TP are the correctly classified samples as positive when it is in fact positive. TN the correctly classified samples as negative when it is in fact negative. FP are the incorrectly classified samples as positive while it is in fact negative. FN are the incorrectly classified samples as negative while it is in fact positive.

4.1.3 Population Size

The increased population size in this thesis shows better results. With a large population size, the genetic algorithm searches the solution space more thoroughly, thereby reducing the chance that the algorithm will fall into a

4.1.2 Fitness Function

Our fitness function is based on the validation process of the solution. We used one fold cross validation technique, a random 1/3 of the data was used for testing and the rest of data used for training SVM with the resulting string kernel. The individual has been processed before the evaluation process. Algorithm 3 shows the individual processing and Algorithm 4 demonstrates the fitness procedure that was used to evaluate the individual. The fitness of the solution can be measured by the multiplying the sensitivity by the specificity according to the following equation:

$$Fitness = Sensitivity \times Specificity \tag{4.3}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{4.4}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{4.5}$$

Where TP are the correctly classified samples as positive when it is in fact positive. TN the correctly classified samples as negative when it is in fact negative. FP are the incorrectly classified samples as positive while it is in fact negative. FN are the incorrectly classified samples as negative while it is in fact positive.

4.1.3 Population Size

The increased population size in this thesis shows better results. With a large population size, the genetic algorithm searches the solution space more thoroughly, thereby reducing the chance that the algorithm will fall into a

Algorithm 3 Individual Preprocessing

begin

1. read the individual from GP to be formulated.
2. read kernel matrices for the training and testing data.
3. substitute the suitable matrices.
4. multiply each sub-kernel with its assigned weight.
5. sum the kernel matrices.
6. produce the combined kernel.
7. go to fitness function to evaluate kernel.

end

Algorithm 4 Fitness Function

begin

1. read combined kernel matrix related to training data.
2. send combined kernel to SVM.
3. train SVM.
4. read combined kernel matrix related to testing data.
5. SVM prediction.
6. calculate performance measures.
7. multiply sensitivity by specificity to calculate fitness.
8. send fitness value to GP for decision.

end

Algorithm 3 Individual Preprocessing

begin

1. read the individual from GP to be formulated.
2. read kernel matrices for the training and testing data.
3. substitute the suitable matrices.
4. multiply each sub-kernel with its assigned weight.
5. sum the kernel matrices.
6. produce the combined kernel.
7. go to fitness function to evaluate kernel.

end

Algorithm 4 Fitness Function

begin

1. read combined kernel matrix related to training data.
2. send combined kernel to SVM.
3. train SVM.
4. read combined kernel matrix related to testing data.
5. SVM prediction.
6. calculate performance measures.
7. multiply sensitivity by specificity to
8. send fitness value to GP for decision

end

local minimum. However, a large population size also causes the algorithm to run more slowly. Population size in the range of 100 to 150 individual was used in the testing of results.

4.1.4 Crossover to Mutation Ratio

At the beginning we use a crossover fraction of 0.5, that we give the population the same percent of being selected for crossover or mutation. However, a crossover fraction of 0.9 is found to be more appropriate for the domain of the work here. This fraction was optimized during the evolution process in GP. The rest of the individuals in the population children are mutated using the mutation function explained above.

4.1.5 Selection Criteria

The selection of parents for reproduction is done using *Lexicographic Parsimony Pressure Tournament* sampling method. Random number of individuals are chosen from the population and the best of them is chosen. However, if two individuals are equally fit, the shortest one (the tree with less nodes) is chosen as the best. This sampling technique has shown to be an effective tool that controls bloat in different types of problems. Other sampling methods are also tested to see which one produces the best kernel, such as tournament, sus and rollete.

4.1.6 Stopping Criteria

Generally, the evolutionary algorithms are usually programmed to stop when reaching one of the following: a specific number of generations, time limit, fitness limit, and stall time limit. In our experiments, we monitor the con-

vergence in fitness during the generations, thus, we do provide the algorithm with a specific number for generations, that causes the fitness value to stop changing starting from a specific number of generations, As a result, we consider this number as a stopping criteria.

In other words, there exist several choices for the number of generations which are given to GP regarding the evolution process, The fitness value is being tracked while the generations keep varying, and when this fitness value stops to improve or change at a certain number of generations, we consider this value and refer to it as the stopping condition.

4.2 Genetic Algorithm Approach

GA was used as another approach for evolving a new combined string kernel. Two methods were adopted to crawl toward our goal, one based on representing the individual as a string of floating point numbers, and the other based on representing the individual as binary string. The following two subsections introduces these methods.

4.2.1 Floating Point Representation

The individual of GA was represented as a string of ten floating point numbers, each number in the string represented the kernel weight. GA searches for the best set of weights. Then the fittest solution consists of ten kernels, each kernel is given the suitable weight. A sample individual is given in the following example:

$$Chromosome1: \quad [0.1\ 0.3\ 0.05\ 0.03\ 0.02\ 0.2\ 0\ 0.06\ 0.04\ 0.2]$$

Then the solution of our GA would be as in the following example:

$$K = 0.1 \times k_1 + 0.3 \times k_2 + 0.05 \times k_3 + 0.03 \times k_4 + 0.02 \times k_5 + 0.2 \times k_6 + 0.06 \times k_8 + 0.04 \times k_9 + 0.2 \times k_{10} \quad (4.6)$$

where k_i are the ten string kernels that was used to evolve the combined kernel.

Each individual was evaluated using the fitness function, after it was processed to produce the kernel matrix. Then the kernel matrix was used a long with SVM to predict the signal peptide sequences. The fittest individual is the one who has the best classification performance. The fitness of the solution can be measured by the multiplying the sensitivity by the specificity as we used it previously in GP in Equation 4.1 .

As we see here in GA we evolved a new string kernel that was represented as a weighted sum of the available ten kernels, by getting the optimized weights from the evolution process of GA. However, this procedure cannot optimize either the kernel parameters, nor the SVM hyperparameter.

4.2.2 Binary String Representation

In this type of representation the GA individual is a binary string consists of ten digits, each digit implies the presence of a kernel in the solution. Zero digit in the i^{th} location means that the k_i is not included in the resulted kernel, one digit means that it is included. A sample individual is given in the following example:

$$Chromosome1: \quad [1\,1\,0\,1\,0\,0\,0\,1\,0\,1]$$

Then the solution would be as in the following example:

$$K = k_1 + k_2 + k_4 + k_8 + k_{10} \quad (4.7)$$

4.3 Implementation

For implementation of GP, we used the GPLAB library [49], MATLAB environment [30] under LINUX for GA. Shogun toolbox [51] was used for SVM and kernel implementation.

Chapter 5

Experiments and Results

This chapter is divided into three sections. In the first section, we introduce a description for the benchmark data. Two benchmark data were used, Signal Peptide and MHC. Then, in the second section, we introduce the results generated from the signal peptide benchmark, including the two evolutionary methods. We look at a specific data set and try to find out how our new GP evolved kernel achieves its good performance. Then we introduce the comparisons between the evolved kernel and other ten string kernels. We also introduce the results from the GA method. In section three, we introduce the results generated from the MHC benchmark data, using the GP evolved kernel. In section four, time comparison between our evolved kernel and other single string kernels in classification is introduced, and the actual time needed to evolve the kernel is also introduced.

Evaluation of the performance is based on the ROC scores that the classifiers achieve on predicting the signal peptide problem and predicting the MHC problem. We present the comparative results based on different performance measures such as classification accuracy, sensitivity, specificity,positive predictive value(PPV), negative predictive value (NPV).

5.1 Benchmark data

Two benchmark datasets are used: the signal peptides and the MHC binding peptides. The first benchmark dataset are a set of protein signal peptides that is used to train the SVM using our evolved kernel, then to be able to classify a protein sequence as a secreted protein or non-secreted protein. The second benchmark dataset are a set of MHC peptides that is used to train the SVM along with the new evolved kernel to distinguish binder peptide from non binder one's.

Our evolved kernel is used in solving protein classification problems. The proteins are the most versatile macromolecules in living systems and serve crucial functions in essentially all biological processes. They transport and store other molecules such as oxygen, provide mechanical support and immune protection, generate movement, transmit nerve impulses, and they control growth and differentiation [22].

Proteins are linear polymers built of from a Repertoire of 20 monomer units called amino acids, These molecules have a common core and differ in only one part called the side chain, which gives the amino acids their specific chemical properties [22]. A description for these two bench marks is introduced here.

5.1.1 Signal Peptide benchmark

Signal peptide is a short (3-60 amino acids long) sequence chain that directs the transport of a protein. It is the pre-sequence that targets proteins that are produced in the cytoplasm to other organelles, such as mitochondria, chloroplasts and apicoplasts, through the secretory pathway [22]. Signal sequence is often located in the N-terminal part in the protein and cleaved off

by an extracellular signal peptidase while the protein is transferred through the membrane. Figure 5.1 illustrates the presence of signal peptide in the N-terminal part of the protein, and shows how the proteins that have signal peptides are secreted through the secretory pathway. Secreted proteins typi-

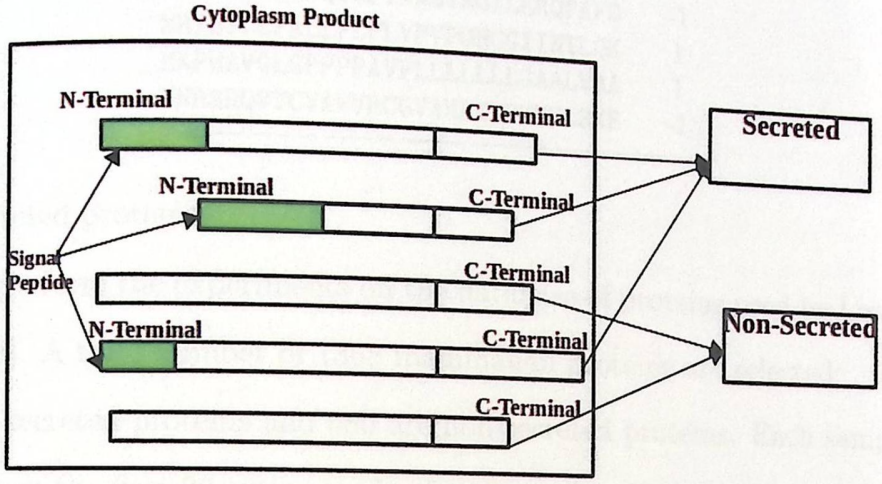


Figure 5.1: Secreted vs Non-Secreted proteins, the cytoplasm produces a protein sequences, some of them have an N-terminal(signal peptide) and C-terminal, others have only C-terminal, the proteins that have signal peptide is secreted outside the cell, while proteins with no signal peptide are not secreted

cally have an N-terminal hydrophobic signal sequence which facilitates their translocation into the endoplasmic reticulum (ER). A signal peptide guides the targeting of secretory proteins to the correct subcellular compartments in the cell, it is often present on the N-terminus of transmembrane proteins.

Signal peptides do not have unique sequences, and their biological characterizations do not provide an accurate enough classification rule. So pattern recognition algorithms are appropriate for this problem since there exists a large set of examples from which to infer a set of rules which discriminate between two patterns, signal peptides vs. non-signal peptides.

We use mammalian proteins to be our first benchmark to evaluate our new evolved kernel along with SVM, by differentiating secreted proteins from

protein sequence	label
MMKILVCLPLLTLYAGCVYGIATGNPVSRL	1
MMSAMKTEFLCVLLLCGAVFTSPSQETYRR	1
MAVWLAQWLGPLLLVSLWGLLAPASLLRRL	1
MHTGGETSACKPSSVRLAPSFSFHAAGLQM	-1
MLRTESCRPRSPAGQVAAASPLLLLLLLLA	1
MLGTVKMEGHETSDWNSYYADTQEAYSSVP	-1
MNDFGIKNMDQVAPVANSYRGTLKRQPAFD	-1
MRIHYLLFALLFLFLVPVPGHGGIINTLQK	1
MKPWAVGLGPPPPAVPLLLLLLLGAALVRA	1
MNRSRQVTCVAWVRCGVAKETPDKVELSKE	-1

non secreted proteins.

We perform the experiments on the database of proteins used by Lezheng et al [63]. A total number of 1365 mammalian proteins are selected: 685 are secreted proteins and 680 are non-secreted proteins. Each sample is taken from the first 30 amino acids of the protein. An example of the signal peptide sequences is given here:

5.1.2 MHC benchmark

Major Histocompatibility Complex (MHC) molecules, are active component to the development of the immune response to pathogens [5]. These molecules act as receptors for peptides (short sequence of amino acids) derived from foreign antigens as well as self peptides and enable the long-term display of antigens on the cell surface [15]. Figure 5.2 illustrates the binding and non-binding peptides, the binding peptides are recognized by the immune system, while non-binding ones are not recognized.

Prediction of peptideMHC binding, represents an important goal in bioinformatics, so we use MHC benchmark data with our evolved kernel to classify binder peptides from non-binder one's using SVM.

The peptide dataset is taken from the NetMHCII 2.2 server, the data used in our experiments is DRB1*0101 datasets [20]. An example of MHCII

protein sequence	label
MMKILVCLPLLTLYAGCVYGIATGNPVSRL	1
MMSAMKTEFLCVLLLCGAVFTSPSQETYRR	1
MAVWLAQWLGPLLLVSLWGLLAPASLLRRL	1
MHTGGETSACKPSSVRLAPSFSFHAAGLQM	-1
MLRTESCRPRSPAGQVAAASPLLLLLLLLA	1
MLGTVKMEGHETSDWNSYYADTQEAYSSVP	-1
MNDFGIKNMDQVAPVANSYRGTLKRQPAFD	-1
MRIHYLLFALLFLFLVPVPGHGGIINTLQK	1
MKPWAVGLGPPPPAVPLLLLLLLGAALVRA	1
MNRSRQVTCVAWVRCGVAKETPDKVELSKE	-1

non secreted proteins.

We perform the experiments on the database of proteins used by Lezheng et al [63]. A total number of 1365 mammalian proteins are selected: 685 are secreted proteins and 680 are non-secreted proteins. Each sample is taken from the first 30 amino acids of the protein. An example of the signal peptide sequences is given here:

5.1.2 MHC benchmark

Major Histocompatibility Complex (MHC) molecules, are active component to the development of the immune response to pathogens [5]. These molecules act as receptors for peptides (short sequence of amino acids) derived from foreign antigens as well as self peptides and enable the long-term display of antigens on the cell surface [15]. Figure 5.2 illustrates the binding and non-binding peptides, the binding peptides are recognized by the immune system, while non-binding ones are not recognized.

Prediction of peptideMHC binding, represents an important goal in bioinformatics, so we use MHC benchmark data with our evolved kernel to classify binder peptides from non-binder one's using SVM.

The peptide dataset is taken from the NetMHCII 2.2 server, the data used in our experiments is DRB1*0101 datasets [20]. An example of MHCII

protein sequence	label
MMKILVCLPLLTLYAGCVYGIATGNPVSRL	1
MMSAMKTEFLCVLLLCGAVFTSPSQETYRR	1
MAVWLAQWLGPLLLVSLWGLLAPASLLRRL	1
MHTGGETSACKPSSVRLAPSFSFHAAGLQM	-1
MLRTESCRPRSPAGQVAAASPLLLLLLLLA	1
MLGTVKMEGHETSDWNSYYADTQEAYSSVP	-1
MNDFGIKNMDQVAPVANSYRGTLKRQPAFD	-1
MRIHYLLFALLFLFLVPVPGHGGIINTLQK	1
MKPWAVGLGPPPPAVPLLLLLLLGAALVRA	1
MNRSRQVTCVAWVRCGVAKETPDKVELSKE	-1

non secreted proteins.

We perform the experiments on the database of proteins used by Lezheng et al [63]. A total number of 1365 mammalian proteins are selected: 685 are secreted proteins and 680 are non-secreted proteins. Each sample is taken from the first 30 amino acids of the protein. An example of the signal peptide sequences is given here:

5.1.2 MHC benchmark

Major Histocompatibility Complex (MHC) molecules, are active component to the development of the immune response to pathogens [5]. These molecules act as receptors for peptides (short sequence of amino acids) derived from foreign antigens as well as self peptides and enable the long-term display of antigens on the cell surface [15]. Figure 5.2 illustrates the binding and non-binding peptides, the binding peptides are recognized by the immune system, while non-binding ones are not recognized.

Prediction of peptideMHC binding, represents an important goal in bioinformatics, so we use MHC benchmark data with our evolved kernel to classify binder peptides from non-binder one's using SVM.

The peptide dataset is taken from the NetMHCII 2.2 server, the data used in our experiments is DRB1*0101 datasets [20]. An example of MHCII

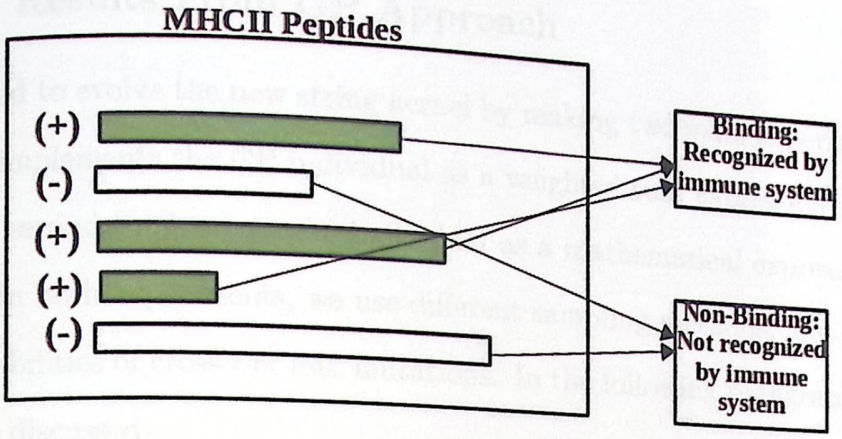


Figure 5.2: MHCII binding and non-binding peptides, the short sequences shown are labeled as (+) or (-), the positive peptides are binded to MHCII and they are recognized by the immune system, while the negative ones are not binded, and so not recognized by immune system

MHC sequence	label
AAEWVLAYMLFTKFF	1
AALNVKRREGMFIDE	-1
AAQPGLTSAVIEALP	-1
ACMLDGGNMLETIKV	1
ACVKDLVSKYLADNE	1
ADLDEILLDGGASDY	-1
ADSEITETYKEGDAV	-1
AEHDRQVLNNLSNCV	1
AEVRSYCYLATVSDL	1
AFKIGLHTEFQTVSF	1

sequences is introduced here:

We use 4794 sequence, divided into 1448 non-binder sequences, and 3346 binder sequence.

5.2 Results from signal peptide benchmark

The signal peptide benchmark is used in testing our new kernel with SVM. Two evolutionary approaches: the GP, and the GA are used to evolve a new kernel on this benchmark. A description for the experiments that are made, will be introduced here.

5.2.1 Results From GP Approach

GP is used to evolve the new string kernel by making two main experiments, the first implements the GP individual as a weighted sum expression of kernels, the second implements the individual as a mathematical expression of kernels. In both experiments, we use different sampling methods, and different probabilities of crossover and mutations. In the following paragraphs the results is discussed.

GP experiment with a weighted-sum individual

We performed our experiment to get the best combination of kernels using a population of 100 kernel trees and 150 generation, each individual is expressed as a weighted sum of string kernels. We applied crossover and mutation operations with 50% probability for each.

Selection of parents for reproduction was done using *'Lexicographic Parsimony Pressure Tournament'* sampling method. This method was explained in (Section 2.3.2). Random number of individuals are chosen from the population and the best of them is chosen. However, if two individuals are equally fit, the shortest one (the tree with less nodes) is chosen as the best. This sampling technique has shown to effectively control bloat in different types of problems [49]. Figure 5.3 illustrates the population diversity during the generations growth, Figure 5.4 depicts the structural complexity, and Figure 5.5 is the resulted kernel tree.

The individual of GP is a kernel tree composed of ten string kernels:

1. Spectrum kernel
2. Weighted Spectrum kernel
3. Fixed-Degree kernel

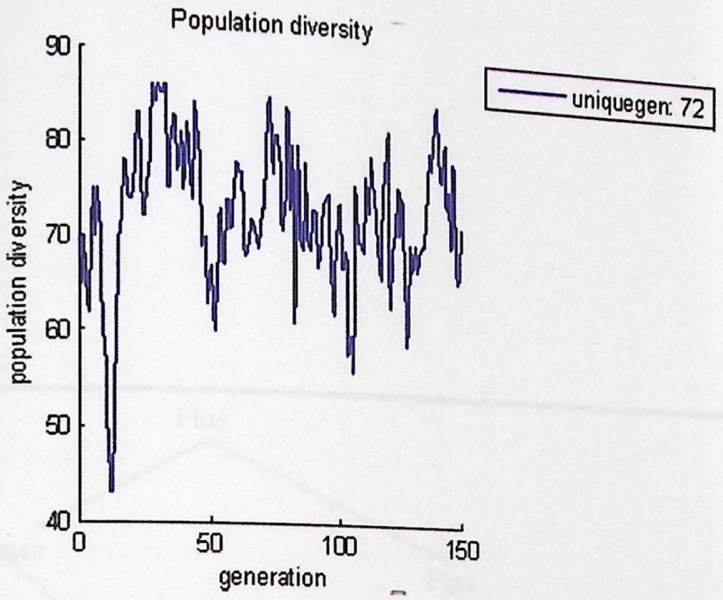


Figure 5.3: Population diversity, it refers to the average distance between individuals in the population, as it is shown, the diversity here is large enough that enables the algorithm to search a larger region of the search space.

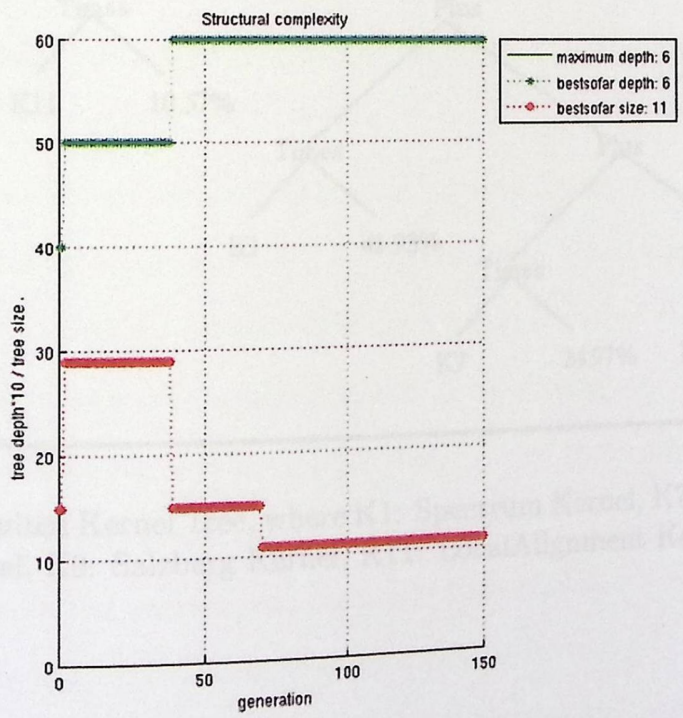


Figure 5.4: The structural complexity of the best evolved tree is illustrated in this figure, the depth of the best tree is 6, and the number of nodes is 11, these numbers shows a small number of nodes(string kernels) produces the best performance.

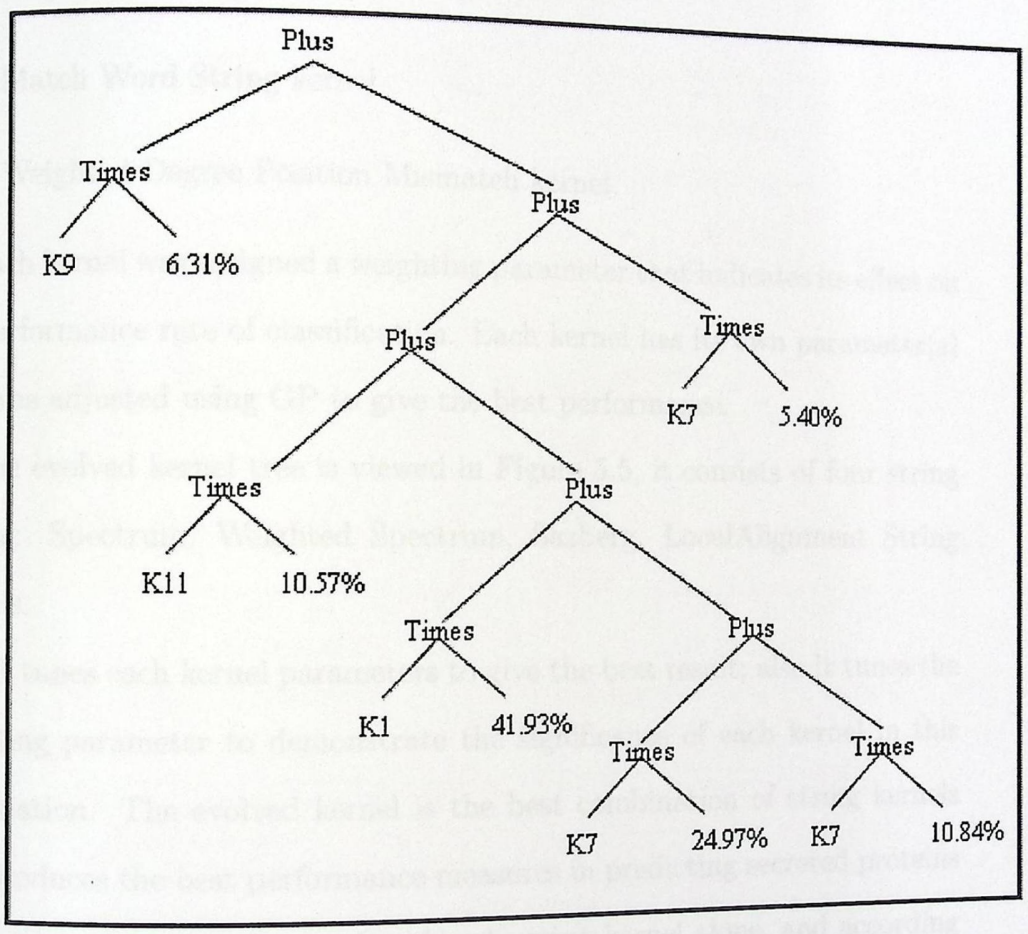


Figure 5.5: Resulted Kernel Tree, where K1: Spectrum Kernel, K7: Weighted Spectrum Kernel, K9: Salzberg Kernel, K11: LocalAlignment Kernel

4. TOP kernel

5. Salzbreg kernel

6. LocalAlignment kernel

7. Localityimproved kernel

8. Polymatch string kernel

9. Match Word String kernel

10. Weighted Degree Position Mismatch kernel.

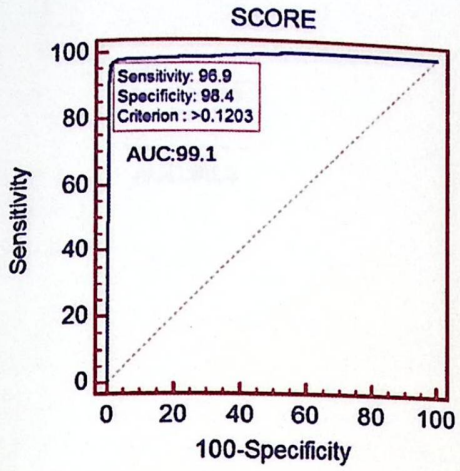
Each kernel was assigned a weighting parameter that indicates its effect on the performance rate of classification. Each kernel has its own parameter(s) that was adjusted using GP to give the best performance.

The evolved kernel tree is viewed in Figure 5.5, it consists of four string kernels: Spectrum, Weighted Spectrum, Sazberg, LocalAlignment String Kernels.

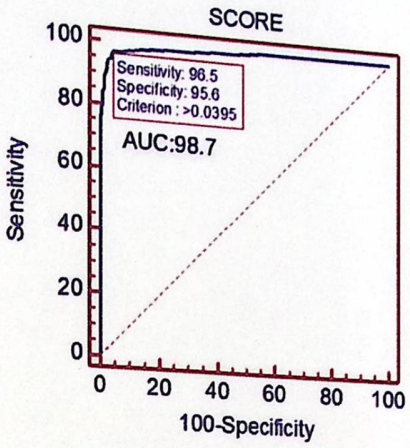
GP tunes each kernel parameters to give the best result; also it tunes the weighting parameter to demonstrate the significance of each kernel in this combination. The evolved kernel is the best combination of string kernels that produces the best performance measures in predicting secreted proteins when comparing with the results of each string kernel alone, and according to our experiments conditions.

Table 5.1 shows the performance measures that obtained when using the new kernel along with SVM compared with the performance of each kernel alone. Three-fold cross validation is used for testing and comparing the results. (Figure 5.6 and Figure 5.7 shows the ROC analysis).

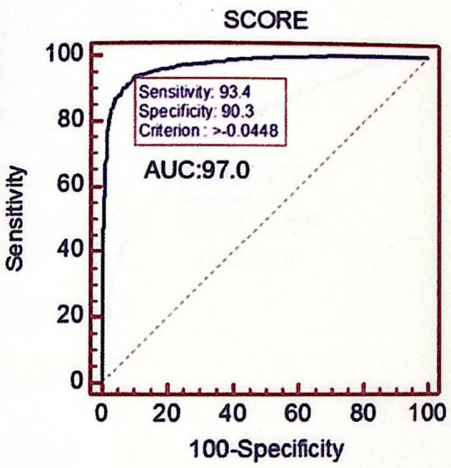
This results show the ability of our model to automatically evolve the suitable kernel with its optimized parameters which outperforms the SVM



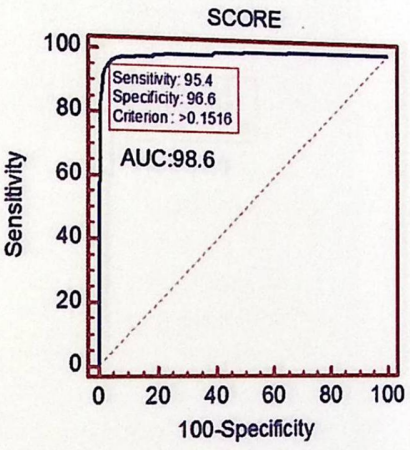
(a) GP evolved kernel



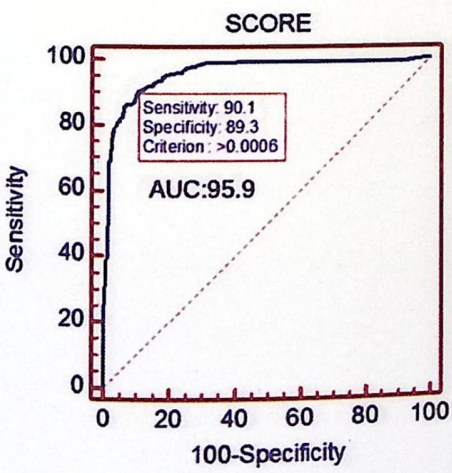
(b) Spectrum kernel



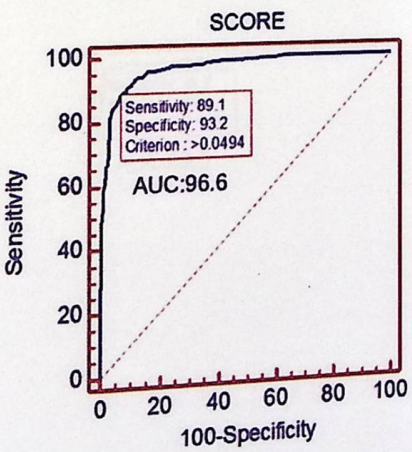
(c) Fixeddegree kernel



(d) Weighteddegreeposition kernel

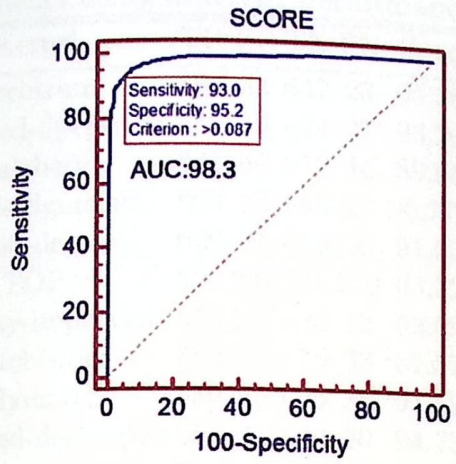


(e) Localityimproved kernel

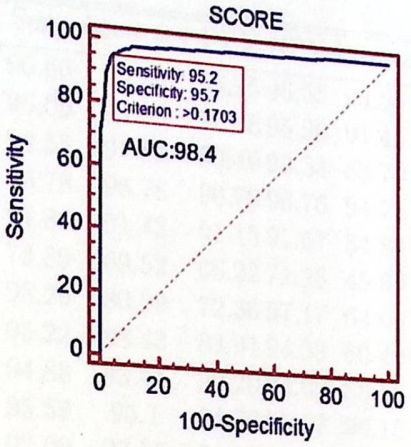


(f) MachWord kernel

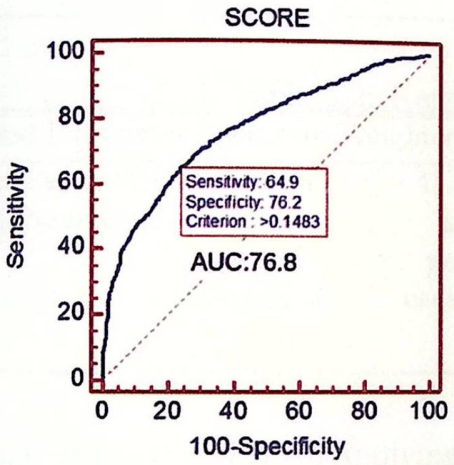
Figure 5.6: Roc Curves resulted from the GP-evolved, Spectrum, Fixeddegree, Wdposition, Localityimproved, Matchword kernels on signal peptide.



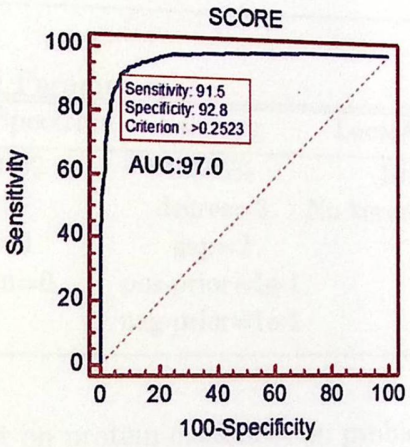
(a) Poymatch kernel



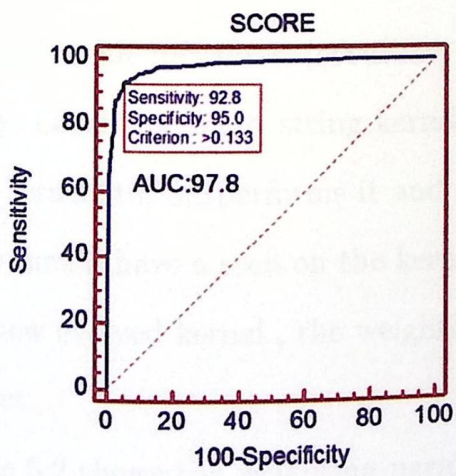
(b) WeightedSpectrum kernel



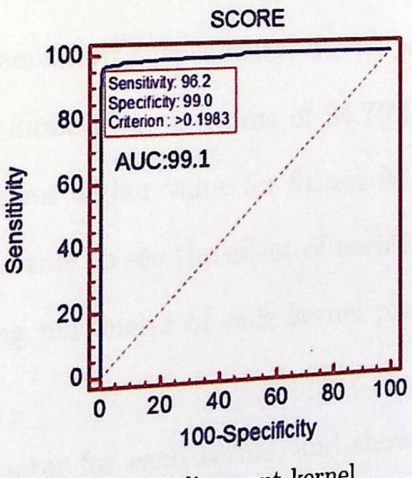
(c) TOP kernel



(d) Salzberg kernel



(e) Weighteddposmismatchkernel



(f) Localalignment kernel

Figure 5.7: Roc Curves resulted from the polymatch, WeightedSpectrum, TOP, Salzberg, Wdpmismatch, Localalignment kernels on signal peptide.

Table 5.1: Results: Comparison of our new evolved kernel with single kernel experiments using different performance measures

Kernel	TP	FP	TN	FN	Spec	Sens	Accuracy	PPV	NPV	Fitness	AUC
Spectrum	662	33	647	23	95.16	96.66	95.90	95.25	96.55	93.55	98.7
Weighted-Spectrum	658	46	634	27	93.20	96.06	94.65	93.48	95.90	91.45	98.4
Salzberg	641	68	612	44	89.95	93.55	91.80	90.49	93.34	83.70	97.0
Local-alignment	663	22	658	22	96.777	96.78	96.78	96.79	96.76	94.75	99.1
Fixed-degree	629	61	619	56	91.03	91.86	91.43	91.15	91.67	84.84	97.0
TOP	506	237	443	179	65.32	73.89	69.52	68.22	71.25	45.69	76.8
Locality-improved	673	257	423	12	62.09	98.26	80.29	72.36	97.17	64.66	95.9
Match-word	652	125	555	33	81.57	95.22	88.43	83.91	94.33	80.40	96.6
Polymatch	650	55	625	35	91.89	94.88	93.41	92.20	94.69	88.56	98.3
Weighted-degreepos	655	36	644	30	94.73	95.59	95.1	94.76	95.57	90.15	98.6
Wdpos-mismatch	637	40	640	48	94.17	92.99	93.55	94.16	93.06	84.90	97.8
New-Evolved-Kernel	667	20	660	18	97.06	97.40	97.22	97.08	97.32	95.68	99.1

Table 5.2: Kernel Parameters

Optimized-Parameter	Spectrum	Weighted-Spectrum	Salzberg	LocalAlignment
Weight-Parameter	41.926%	41.204%	6.305%	10.565%
Kernel-Parameters	k=2	k=2	degree=3	No kernel parameter
	gap=1	gap=0	gap=2	
	usesign=0	usesign=0	pos-prior=1e-1	
			neg-prior=1e-1	

with single string kernel when applying it on protein classification problems. We can see from the table above that our evolved kernel gives the highest percentage in all measures (accuracy, sensitivity, specificity, PPV, NPV), while the LocalAlignment string kernel alone gives a fitness of 94.75%, our evolved kernel still outperforms it and gives higher value for fitness 95.68%. Here we should have a seen on the kernel tree to see the effect of each kernel on the new evolved kernel , the weighting parameter of each kernel presents this effect.

Table 5.2 shows the weighting parameter for each kernel, and shows the different kernel parameters that our method optimized them. Spectrum kernel participates with 41.926%, WeightedSpectrum kernel participates with 41.204%, Salzberg kernel participates with 6.305%, LocalAlignment kernel

participates with 10.565%. In spite of the power of the LocalAlignment string kernel alone, it does not take a big role in our combined kernel. Also our model optimized the SVM parameter C, it tuned this parameter to be C=100 that gives the best performance.

Experiment with mathematical expression individuals

The individual of the GP here was a mathematical expression that preserves the closure properties. A set of mathematical functions that preserves the closure properties can be used such as summation, multiplication, exponential and others. However not all of them could be used in our experiments, such as exponential function. This because we deal with large size kernel matrices, and the solution that contains many exponential functions for these matrices needs larger size of memory, larger space, faster processors than the available ones.

So the multiplication function, with some limitation on the tree depth and node size, was used in addition to the summation function. However the GP chose the best individual the contains only the summation function. The best individual can seen as the following equation:

$$K_1 \times 72.6 + K_3 \times 25.2 + K_{11} \times 2.2 \tag{5.1}$$

Where K_1 is the Spectrum kernel, K_3 is the Weighted-degree-position kernel, and K_{11} is the Localalignment kernel.

In this experiment the crossover of fraction 0.95 was chosen by GP to be the best ratio. 150 generations and a population of size 100 were used to evolve the kernel. The evolved kernel was used with SVM to classify secreted proteins, and the resulted ROC curve can be viewed in Figure 5.8 .

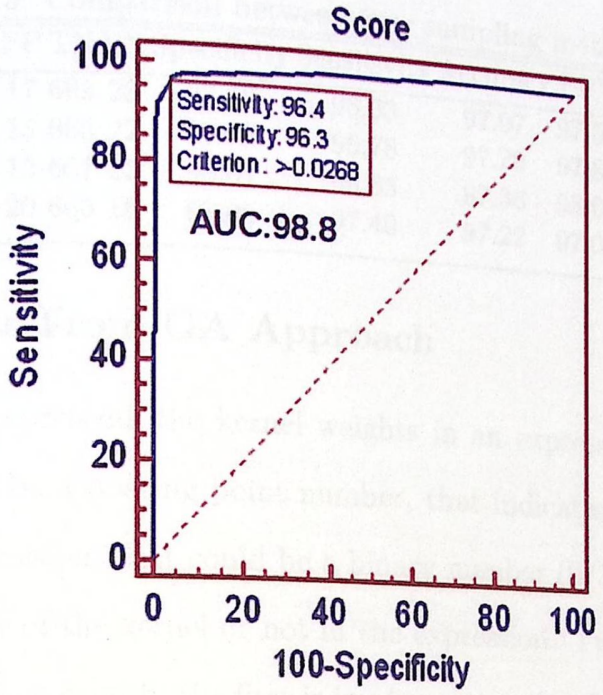


Figure 5.8: ROC curve for evolved kernel when individual is mathematical expression

These Results confirms that we could obtain the same performance as the Localalignment kernel using simple string kernels, as we notice that from the high value for the Spectrum kernel weight. Also we should not forget that this method in general facilitate the process of selecting the suitable string kernel for a specific problem, and it optimizes the different kernel and SVM parameters.

Comparison between different sampling methods

There are different sampling methods that can be used during the evolutionary process, Table 5.3 shows the comparison results between four sampling methods, when we used a crossover fraction of 0.8, and using 150 generations on 100 population size. The results shows that the Lexicographic parsimony pressure method is the best sampling method in this evolution problem.

Table 5.3: Comparison between four sampling methods

Sampling	TP	FP	TN	FN	Specificity	Sensitivity	Accuracy	PPV	NPV	Fitness
Roulette	662	17	663	23	97.49	96.63	97.07	97.50	96.65	94.20
Tournament	663	15	665	22	97.78	96.78	97.29	97.80	96.80	94.63
Sus	662	13	667	23	98.07	96.63	97.36	98.08	96.67	94.77
Lexicographic	667	20	660	18	97.06	97.40	97.22	97.08	97.32	95.68

5.2.2 Results From GA Approach

GA chromosome represents the kernel weights in an expression of kernels. This weight could be a floating point number, that indicates the kernel influence in the expression, or it could be a binary number (0/1) that informs about the presence of the kernel or not in the expression. Two experiments are made using GA approach, the first is implementing the chromosome as a floating point numbers, the second is implementing the chromosome as set of binaries. The results from these two experiments is discussed in the following subsections.

Floating Point Chromosome Results

We performed this experiment to get the best weighted combination of all the available kernels. All the string kernels are included here. The resulted GA chromosome can be expressed as the following:

Best chromosome: 0.1671, 0.0152, 0.0730, 0.0001, 0.0893, 0.0340, 0.2325, 0.0029, 0.0269, 0.0139, 0.3452

Each floating point number in this solution is a weight for the corresponding string kernel. Applying the resulted kernel a long with SVM to predict signal peptide and to distinguish between secreted and non secreted proteins, a population of 100 individual represented a weighted combination of string kernels, and 150 generation was used. Figure 5.9 shows the fitness behavior

during generations. The resulted kernel could be viewed as the following:

$$16.71 \times K_1 + 1.52 \times K_2 + 7.3 \times K_3 + 0.01 \times K_4 + 8.93 \times K_5 + 3.4 \times K_6 + 23.25 \times K_7 + 0.29 \times K_8 + 2.69 \times K_9 + 1.39 \times K_{10} + 34.52 \times K_{11} \tag{5.2}$$

Then the ROC analysis resulted from applying this kernel along with SVM can be viewed in Figure 5.10.

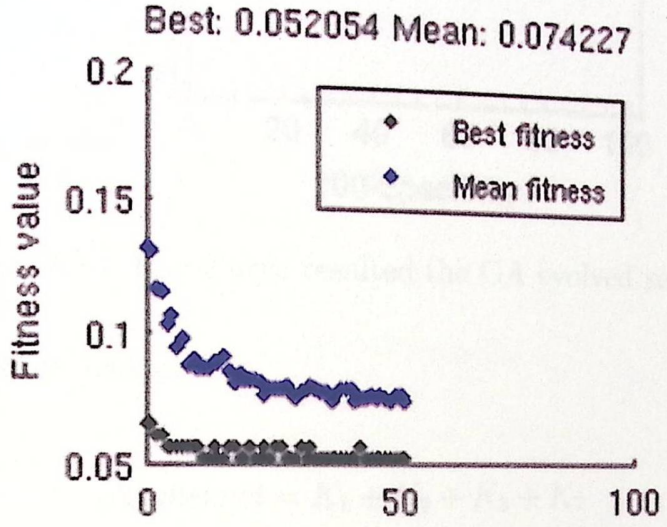


Figure 5.9: Fitness behavior during the GA evolution

0/1 Chromosome Results

We used GA to evolve a new string kernel using another form of expression, the new string kernel can be viewed here as a combination of some string kernels, not all of them as we have seen in the previous implementation for the GA individual. In this experiment the individual is a string of binary digits, each digit implies the presence of the corresponding string kernel.

The resulted chromosome can be viewed as in the following: Best chromosome: 1 1 1 0 0 0 1 0 0 0

the 1's binary digits in this solution implies that the corresponding string kernels are used in the new string kernel expression. So the solution can be

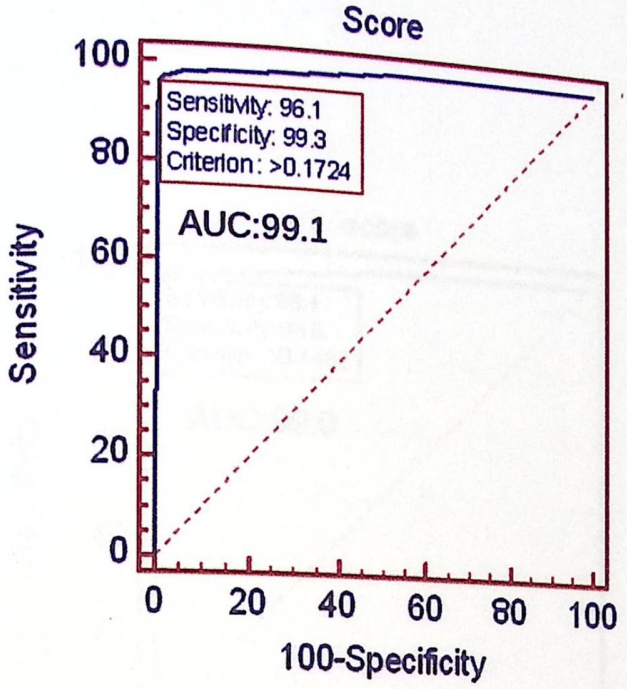


Figure 5.10: Roc Curve resulted the GA evolved kernel

expressed as the following:

$$Newkernel = K_1 + K_2 + K_3 + K_7 \tag{5.3}$$

Where K_1 is the Spectrum kernel, K_2 is the Fixed-degree Kernel, K_3 is the Weighted-degree-position kernel and K_7 is the weighted-spectrum kernel. The new evolved kernel was then used along with SVM to classify mammalian secreted proteins. The results can be seen in Figure 5.11 as a ROC analysis.

A comparison between the GA evolved string kernel, the evolved GP string kernel, and the Localalignment string kernel which is the best among other string kernels, can be viewed in Table 5.4.

From the results shown above, we can see that the GP evolved kernel is the best among all when we look at the accuracy and the value of fitness. While the GA kernel(1) and GA kernel(2) have less performance than GP kernel, they outperforms the Localalignment kernel in accuracy, notifying that GA kernel(1) is a weighted combination of all these kernel, and GA

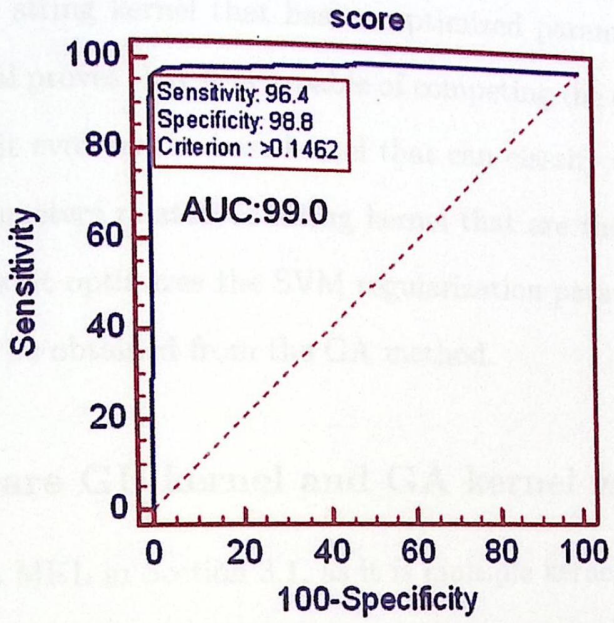


Figure 5.11: Roc Curve resulted from the second GA kernel

Table 5.4: Comparison between GA kernel, GP kernel and Localalignment kernel

Kernel	TP	FP	TN	FN	Specificity	Sensitivity	Accuracy	PPV	NPV	Fitness
Local-alignment	663	22	658	22	96.777	96.78	96.78	96.79	96.76	94.75
GP Kernel	667	20	660	18	97.06	97.40	97.22	97.08	97.32	95.68
GA kernel(1)	663	17	663	22	97.50	96.78	97.14	97.49	96.79	94.36
GA kernel(2)	664	16	664	21	97.64	96.93	97.29	97.65	96.93	94.64

kernel(2) is a summation of four string kernels.

These results confirms that GP kernel is not only the best of them, however it is the only string kernel that has an optimized parameters. So GP approach in general proves that it is capable of competing the existing methods, this because it evolved the best kernel that can classify the dataset, it optimized all parameters related to string kernel that are the basis for the evolved kernel, also it optimizes the SVM regularization parameter (C). All this work can not be obtained from the GA method.

5.2.3 Compare GP kernel and GA kernel with MKL

As we talk a bout MKL in Section 3.1, as it is multiple kernel learning with different optimization techniques. A comparision between MKL and our evolved kernels: GP-kernel and GA kernels, is shown in Table 5.5. These

Table 5.5: Comparison between MKL, GA kernel and GP kernel

Kernel	Specificity	Sensitivity	Accuracy	PPV	NPV	Fitness
MKL	96.44	97.21	96.85	96.54	97.20	93.75
GP Kernel	97.06	97.40	97.22	97.08	97.32	95.68
GA kernel(1)	97.50	96.78	97.14	97.49	96.79	94.36
GA kernel(2)	97.64	96.93	97.29	97.65	96.93	94.64

results shows the powerful of our optimized GP kernel against MKL, when it is measured in all performance measures. Also our GA kernels outperforms MKL in accuarcy, fitness which is sensitivity multiplied by specificity.

5.3 Results from MHC benchmark

The MHC benchmark that is described in Section 5.1.2, is used in testing our new kernel with SVM. GP is used to evolve a new kernel on this bench-

mark. However, GA is not used as in signal peptide experiments, because we realize from the signal peptide prediction results, that GP is better than GA in evolving a string kernel, and more flexible in optimizing the kernels parameters.

A description for the experiments that is made, will be introduced here.

We perform an experiment on GP using 100 generation and 90 population size during the evolution process. The best ratio of the crossover to mutation is optimized during the GP evolution process, 80% for crossover and 20% for mutation. Five experiments are performed on equal size of binder and non-binder data, each experiment is then made in three fold cross validation. A comparison of the average performance measures between our GP kernel on MHC and the single kernels results on MHC can be shown in table 5.6.

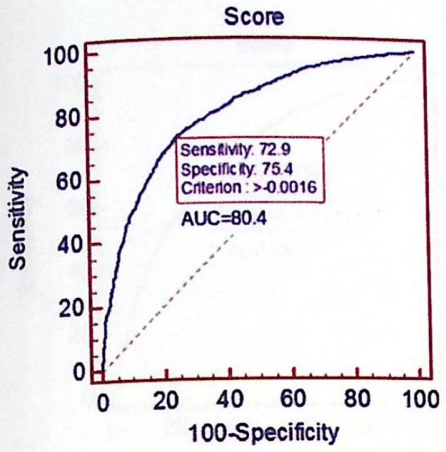
From this table we can see that our evolved kernel outperforms the single kernels in accuracy, sensitivity and specificity, and fitness.

Table 5.6: Average performance measure resulted from GP kernel on MHC

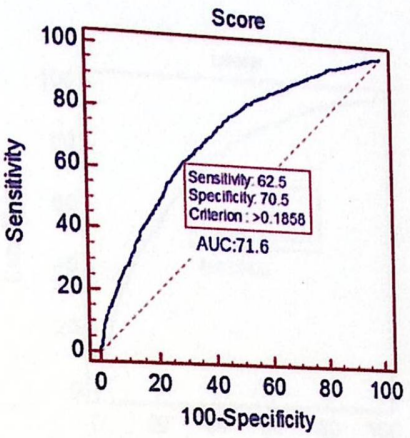
Kernel	Specificity	Sensitivity	Accuracy	PPV	NPV	Fitness	AUC
new-evolved-Kernel	76.95	72.20	73.51	75.89	74.57	56.05	80.40
Spectrum	67.53	66.17	66.66	67.09	66.85	45.86	71.60
Weighted-Spectrum	69.66	67.11	67.95	68.92	68.39	46.67	74.80
Salzberg	87.74	20.19	52.36	62.26	53.96	17.89	59.5
Local-alignment	65.61	65.29	65.39	65.54	65.45	42.77	69.9
Fixed-degree	61.56	61.31	61.41	61.51	61.44	38.07	65.80
TOP	54.38	58.73	56.89	56.35	56.55	32.48	59.00
Locality-improved	91.78	26.37	55.51	76.45	59.06	24.87	68.30
Match-word	66.16	64.63	65.17	65.66	65.40	42.08	70.20
Polymatch	68.38	69.03	68.84	68.61	68.70	46.64	75.00
Weighted-degreepos	66.69	67.47	67.23	66.99	67.08	44.76	73.00
Wdpos-mismatch	72.68	60.75	65.12	69.89	66.72	43.42	69.70

The Roc Curves for GP-kernel, and for the single kernels on MHC data can be shown here:

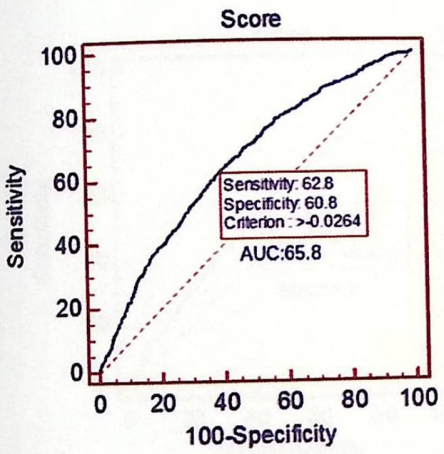
The experiments made on MHC benchmark using, shows the powerful of the evolved kernel against other string kernel. It gives the higher perfor-



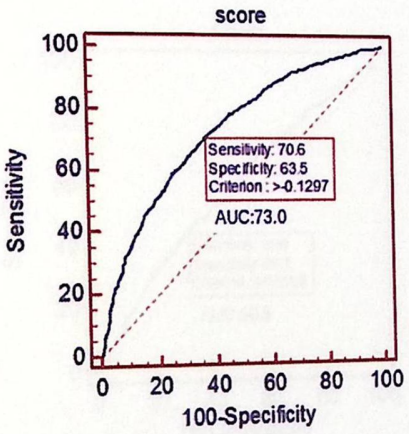
(a) GP evolved kernel



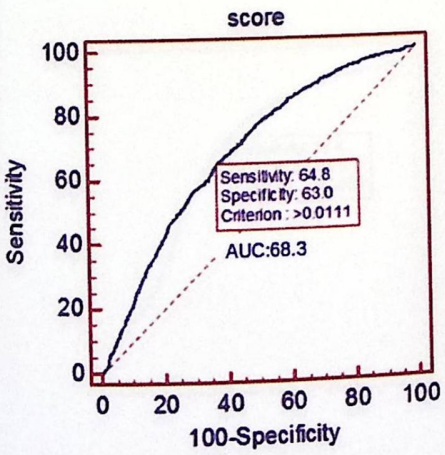
(b) Spectrum kernel



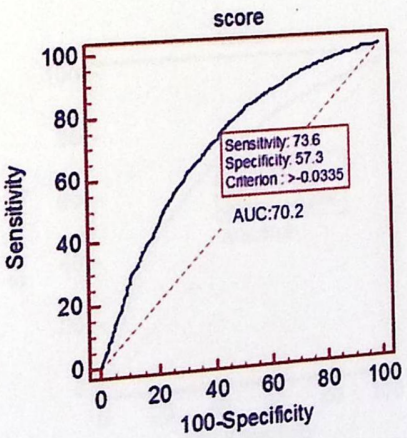
(c) Fixeddegree kernel



(d) Weighteddegreeposition kernel

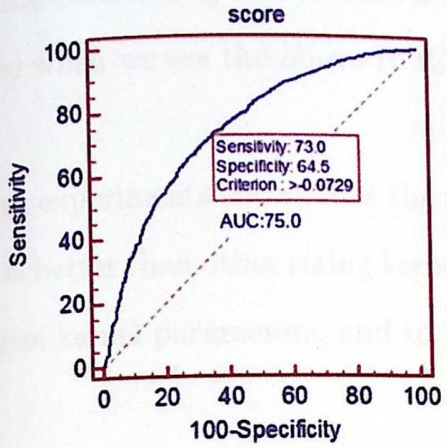


(e) Localityimproved kernel

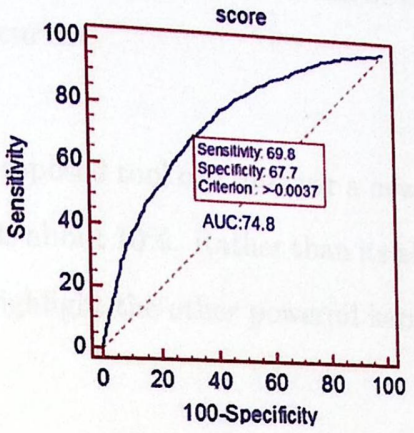


(f) MachWord kernel

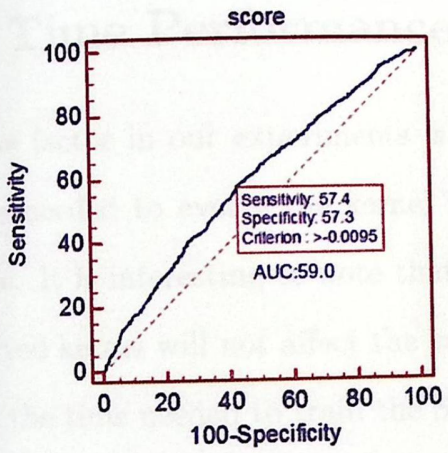
Figure 5.12: Roc Curves resulted from the GP-evolved, Spectrum, Fixed-degree, Weighteddegreeposition, Localityimproved, Matchword kernels on MHC.



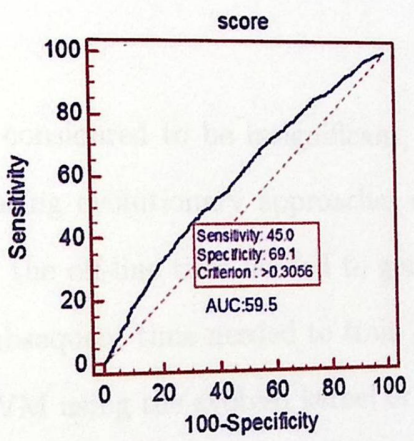
(a) Poymatch kernel



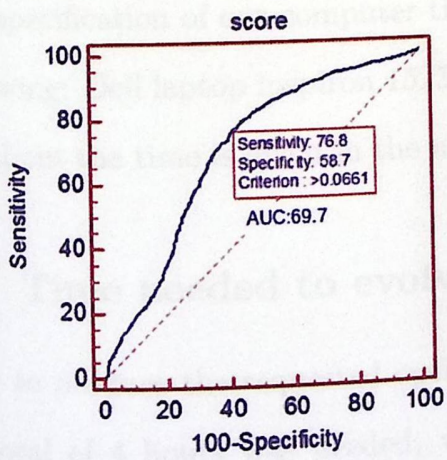
(b) WeightedSpectrum kernel



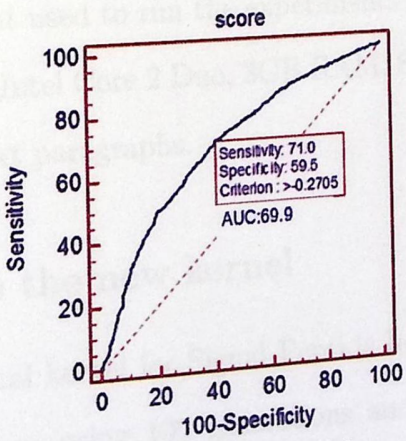
(c) TOP kernel



(d) Salzberg kernel



(e) Weighteddposmismatchkernel



(f) Localalignment kernel

Figure 5.13: Roc Curves resulted from the polymatch, WeightedSpectrum, TOP, Salzberg, Weighteddposmismatch, Localalignment kernels on MHC.

mance measures,for example, its area under curve(AUC) is 80.4, while the AUC in the other string kernels does not exceed seventies. This can be easily concluded when we see the above ROC curves.

So this experiments shows that the proposed tool can discover a new kernel that is better than other string kernels about 10%. Rather than its ability to optimize kernel parameters, and to highlight the other powerful kernels.

5.4 Time Performance

The time factor in our experiments is considered to be insignificant, since the time needed to evolve the kernel using evolutionary approaches is off-line time. It is interesting to note that the off-line time needed to generate the evolved kernel will not affect the subsequent time needed to train SVM, because the time needed to train the SVM using the evolved kernel or using any single kernel is the same.

The specification of our computer that used to run the experiments is as the following: Dell laptop inspiron 1525, Intel Core 2 Duo, 3GB RAM. Some details about the time is given in the next paragraphs.

5.4.1 Time needed to evolve the new kernel

In order to produce the requested optimal kernel for Signal Peptide benchmark, total of 4 hours was needed, when using 100 generations and 150 population size, under the same condition of the computer. On the other hand, to evolve the kernel for MHC benchmark, we found that the a total of 7 hours was needed, at the same conditions.

As we see that the time needed to evolve kernel for MHC benchmark is larger than the time needed to evolve kernel for signal peptide, because the number of samples in MHC is 4794, it is larger than the signal peptide samples that are 1365 sequence. And as we know that the kernel matrix dimension is the number of samples squared.

5.4.2 Comparison time between our evolved kernel and other string kernels

In our experiments we compare the classification performance of the evolved string kernel with performance of the single kernels, so when we compare the time, we would say that the time needed to train the SVM using either our evolved kernel or single kernels is the same. Since the kernel matrix dimension depends on the number of samples. Our evolved kernel is applied on the same number of sample, also it is a result of applying mathematical operations on the matrices, and as we know the summation of 2 matrices with the same dimension resulted in a matrix with the same dimension.

Chapter 6

Conclusion and Future Work

This thesis has proposed a novel approach that automatically generates the optimized string kernel. The new evolved kernel combines set of string kernels as an optimized expression, with optimized parameters. The proposed model uses two evolutionary approaches to evolve the suitable string kernel for biological problems, the GP approach, and the GA approach. The results show that the evolved kernel with SVM is capable of beating the performance of available methods.

This model solves the problem of kernel selection for SVM and optimizes the kernel parameters and SVM regularization parameter. This model has the potential to discover new string kernels for particular problem. The new evolved kernel tested on a dataset of secreted and non-secreted proteins, it has significantly better ROC scores than other string kernels.

This Tool has the ability to select the powerful kernels, that can't be discovered using other tools.

The evolved kernel is based on ten different string kernels: Spectrum kernel, Weighted Spectrum kernel, Fixed-Degree kernel, TOP kernel, Salzbreg kernel, LocalAlignment kernel, Localityimproved kernel, Polymatch string

kernel, Match Word String kernel, Weighted Degree Position Mismatch kernel. While these kernels can be used to get good results in classification, the combination of them using our model surely can produce better results.

Our model based on GP approach and GA approach not only produce the best string kernel from the combination of kernels, but also it facilitates kernel parameter selection mission, introduces the best parameters for each kernel and the best regularization parameter for SVM.

Our experiments proves that we are able to discover new string kernels that can give results either equal or more than the results from the available kernels.

The new evolved kernel, outperforms the MKL that combines a set of kernels and optimize them using different optimization techniques.

The results of our experiments in evolving a string kernel, shows that GP approach is better than GA approach.

The experiments made on MHC benchmark, shows the powerful of the evolved kernel against the other string kernel. Also the results from using the evolved kernel with SVM in classifying MHC benchmark, is comparable with the recent researches results.

The results of our evolved kernel using either GA approach or GP approach, involves that all performance measures and fitness was not significantly converge. In other words, the fitness starts from high value and increased slightly during the generations, this can be seen as a result of that all the string kernels that were used as a basis for our evolved kernel were powerful kernels, each one can act well in classifying specific data. This means that the results could be improved by considering larger number of string kernels to be used as a basic for combination, or considering more data specific kernel such as physio-chemical properties kernels.

One of the main difficulties that we faced in developing our model is that the memory and speed limitations. This because the testing data was biological sequences consists of large number of samples with long length, and the string kernel produces a kernel matrix with a squared dimension of the number of samples, rather than the time needed by string kernel process itself. So another opportunity for improving this model is by enhancing the implementation for GP, GA or the string kernels, using algorithms enhancements or parallelism.

It is still possible to improve this model by evolving the string kernel using other evolutionary approaches such as Particle Swarm Optimization (PSO) [53] and others. It could therefore be an idea to use boosting for combining and optimizing string kernels.

Bibliography

[1] H. Ahn, K. Lee, and K. Kim. Global optimization of support vector machine using genetic algorithms for bankruptcy prediction. *ICONIP'06 Proceedings of the 13th international conference on Neural information processing - Volume Part III*, 2006.

[2] SF. Altschul and EV. Koonin. Iterated profile searches with psi-blast–a tool for discovery in protein databases. *Trends Biochem Sci*, pages 444–447, 1998.

[3] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research2*, pages 125–137, 2001.

[4] A. Ben-Hur, C. Ong, S. Sonnenburg, B. Scholkoph, and G. Ratsch. Support vector machines and kernels for computational biology. *PLoS Computational Biology 4(10*, pages 1–10, 2008.

[5] ST. Chang, D. Ghosh, DE. Kirschner, and JJ. Linderman. Peptide length-based prediction of peptidemhc class ii binding. *Bioinformatics, Volume 22 Issue 22*, page 27612767, 2006.

[6] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Mach. Learn*, pages 131–159, 2002.

[7] C. Coello, G. Lamont, and D. Veldhuizen. Evolutionary algorithms for solving multi-objective problems.

[8] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. *NIPS, MIT Press*, pages 537–544, 2002.

[9] L. Diosan, A. Rogosan, and JP. Pecuchet. Optimizing multiple kernels for svm by genetic programming. *EvoCOP'08 Proceedings of the 8th European conference on Evolutionary computation in combinatorial optimization, Springer-Verlag Berlin, Heidelberg*, 2008.

[10] JP. Egan. *Signal detection theory and ROC analysis*. Series in Cognition and Perception. Academic Press, New York, 1975.

[11] T. Fawcett. An introduction to roc analysis. *Pattern Recognition*, page 861874, 2006.

[12] F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameterss. *in Proceedings of the 12th European Symposium on Artificial Neural Network*, page 519524, 2004.

[13] F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, pages 107–117, 2005.

[14] SR. Gunn. Support vector machines for classification and regression. *Technical Report,UNIVERSITY OF SOUTHAMPTON*, 1998.

[15] J. Hammer, E. Bono, F. Gallazzi, C. Belunis, Z. Nagy, and F. Sinigaglia. Precise prediction of major histocompatibility complex class ii-peptide interaction based on peptide side chain scanning. *The Journal of experimental medicine*, pages 2353–2358, 1994.

[16] T. Handsat. Protein remote homology detection using motifs made with genetic programming. *MA-thesis,Norwegian University of Science and Technology*, 2006.

[17] D. Haussler. Convolution kernels on discrete structures. *Technical Report UCS-CRL-99-10*, 1999.

[18] T. Howley and M. Madden. The genetic kernel support vector machine description and evaluation. *Artificial Intelligence Review archive Volume 24 Issue 3-4, Kluwer Academic Publishers Norwell, MA, USA*, 2005.

[19] T. Howley and M. Madden. An evolutionary approach to automatic kernel construction. *ICANN'06 Proceedings of the 16th international conference on Artificial Neural Networks - Volume Part II, Springer-Verlag Berlin, Heidelberg*, 2006.

[20] C. Immunological Bioinformatics. Center for biological sequence analysis. `http://www.cbs.dtu.dk/suppl/immunology/NetMHCII-2.0.php`.

[21] T. Jaakkola, M. Diekhans, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. *In Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999.

[22] B. Jeremy, T. John, and S. Lubert. *Biochemistry*. W.H. Freeman and Company, 5 edition, 2002.

[23] L. Kall, A. Krogh, and ELL. Sonnhammer. A combined transmembrane topology and signal peptide prediction method. *Mol Biol*, 2004.

[24] JR. Koza. Genetic programming: On the programming of computers by means of natural selection. *MIT press, Cambridge, MA, USA*, 1992.

[25] G. Lanckriet, N. Cristianini, L. Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semi-definite programming. *Machine Learning Research*, 5:2772, 2004.

[26] C. Leslie and E. Eskin. The spectrum kernel a string kernel for svm protein classification. *Pacific Symposium On Biocomputing, Volume: 575, Issue: 50*, pages 564–575, 2002.

[27] C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics, Vol. 20, No. 4.(1 March 2004)*, pages 467–476, 2004.

[28] C. Leslie, E. Eskin, J. Weston, and W. Noble. Mismatch string kernels for svm protein classification. *In Advances in Neural Information Processing Systems 15*, pages 1417–1424, 2003.

[29] H. Li and T. , Jiang. A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas. *Proc. Eighth Ann. Intl Conf. Computational Molecular Biology*, 2004.

[30] MathWorks-Inc. Matlab version 7.6.0. `http://www.mathworks.com/index.html`, 2010. Natick Massachusetts.

[31] B. Mersch, T. Glasmachers, P. Meinicke, and C. Igel. Evolutionary optimization of sequence kernels for detection of bacterial gene starts. *ICANN'06 Proceedings of the 16th international conference on Artificial Neural Networks - Volume Part II, Springer-Verlag Berlin, Heidelberg*, 2006.

[32] I. Methasate and T. Theeramunkong. A family-based evolutional approach for kernel tree selection in svms. *PAKDD '09 Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining.*

[33] I. Methasate and T. Theeramunkong. Kernel trees for support vector machines. *IEICE Transactions*, pages 1550–1556, 2007.

[34] Z. Michalewics. *Genetic Algorithms+Data Structures=Evolution Programs.* Springer Berlin Heidelberg New York, 3 edition, 1996.

[35] N. Mukherjee and S. Mukherjee. Predicting signal peptides with support vector machines. *In Pattern Recognition with Support Vector Machines*, 2388, 2002.

[36] H. Nielsen and G. Brunak, S.and von-Heijne. Machine learning approaches for the prediction of signal peptides and other protein sorting signals. *Protein Eng.*, 1999.

[37] H. Nielsen, S. Brunak, J. Engelbrecht, and G. Von-Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Prot. Eng.*, 1997.

[38] H. Nielsen, J. Engelbrecht, S. Brunak, and G. Von-Heijne. A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Int. J. Neural Syst.*, page 581599, 1997.

[39] StatTools Home Page. Receiver operator characteristics explained. `http://www.stattools.net/ROCs_Exp.php`, 2011.

[40] T. Phienthrakul and B. Kijsirikul. Evolutionary strategies for multi-scale radial basis function kernels in support vector machines. *in proceeding of 2005 conference on genetic and evolutionary combutations, ACM Press, New York*, pages 905–911, 2005.

[41] T. Phienthrakul and B. Kijsirikul. Gpes:an algorithm for evolving hyprid kernel functions of support vector machine. *IEEE Congress on Evolutionary Computations,IEEE Press, Los Alamitos*, pages 2636–2643, 2007.

[42] D. Plewczynski, L. Slabinski, K. Ginalski, and L. Rychlewski. Prediction of signal peptides in protein sequences by neural networks. *Acta Biochim Pol*, 2008.

[43] G. Ratsch and S. Sonnenburg. Accurate splice site detection for caenorhabditis elegans. *In Schlkopf,B., Tsuda,K. and Vert,J.P. (eds.), Kernel Methods in Computational Biology, MIT Press, Cambridge*, pages 277 – 298, 2004.

[44] H. Saigo, J. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, pages 1682–1689, 2004.

[45] J. Salomon and DR. Flower. Predicting class ii mhc-peptide binding: a kernel based approach using similarity scores. *BMC Bioinformatics*, 2006.

[46] SL. Salzberg. A method for identifying splice sites and translational start sites in eukaryotic mrna. *Computer Applications in the Biosciences*, page 365 376, 1997.

[47] B. Scholkopf and AJ. Smola. Learning with kernels: Support vector machines, regularization, optimization, and beyond. *MIT Press, Cambridge, MA, USA*, 2001.

[48] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[49] S. Silva. A genetic programming toolbox for matlab. `gplab.sourceforge.net`, 2009.

[50] TF. Smith and MS. Waterman. Identification of common molecular subsequences. *J Mol Biol*, pages 195–197, 1981.

[51] S. Sonnenburg, G. Raetsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. Bona, A. Binder, C. Gehl, and V. Franc. The shogun machine learning toolbox. *Journal of Machine Learning Research*, pages 1799 – 1802, 2010.

[52] S. Sonnenburg, G. Raetsch, C. Schaefer, and B. Scholkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[53] B. Souza, A. Carvalho, R. Calvo, and R. Ishii. Multiclass svm model selection using particle swarm optimization. *Hybrid Intelligent Systems 2006 HIS 06 Sixth International Conference(ICHIS6),Issue: 1*, page 31, 2006.

[54] K. Sullivan and S. Luke. Evolving kernels for support vector machine classification. *GECCO '07 Proceedings of the 9th annual conference on Genetic and evolutionary computation , ACM New York, NY, USA*, 2007.

[55] J. Sun and L. Wang. Predicting signal peptides and their cleavage sites using support vector machines and improved position weight matrices. *In Proceedings of the 4th International Conference on Natural Computation, ICNC*, 2008.

[56] JA. Swets, RM. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, pages 283, 8287, 2000.

[57] J. Taylor and N. Cristianini. Kernel methods for pattern analysis. *Cambridge University Press, New York, NY, USA*, 2004.

[58] V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc. New York, NY, USA, 1995.

[59] J. Vert, T. Akutsu, and H. Saigo. Local alignment kernels for biological sequences. in kernel methods in computational biology, B.,Scholkopf and K.,Tsuda and JP.,Vert (Eds.). *MIT Press*, pages 131–154, 2004.

[60] JP. Vert. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. *Pac Symp Biocomput*, pages 649–660, 2002.

[61] L. Wang. *Support Vector Machines: Theory and Applications*, volume 177. Springer Berlin Heidelberg New York, 2005.

[62] M. Wang, J. Yang, and KC. Chou. Using string kernel to predict signal peptide cleavage site based on subsite coupling model. *Amino Acids*, pages 395–402, 2005.

[63] L. Yu, Y. Guo, Z. Zhang, Y. Li, M. Li, G. Li, W. Xion, and Y. Zeng. Secretp a new method for predicting mammalian secreted proteins. *Elsevier, Peptides31*, page 574578, 2010.

[64] XM. Zhao, DS. Huang, and YM. Cheung. A novel hybrid ga/svm system for protein sequences classification. *Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL),Springer-Verlag, UK*, pages 11–16, 2004.

[65] A. Zien, G. Raetsch, S. Mika, T. Schoelkopf, B.and Lengauer, and KR. Mueller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, pages 799 – 807, 2000.

[66] L. Zou, Z. Wang, Y. Wang, and F. Hu. Combined prediction of transmembrane topology and signal peptide of beta-barrel proteins. *using a hidden Markov model and genetic algorithms, Computers in biology and medicine*, 40:621–628, 2010.