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

Protein Sequence Data: Laying the Mathematical Foundations to Derive Novel Descriptors

Submitted by:

Berat Huseyin Alisan Kurar

A Thesis submitted in partial fulfillment of requirements for the degree of Master of Science in Informatics
November, 2015

In the Name of Allāh, the Most Gracious, the Most Merciful

Graduate Advisory Committee:

Dr. Sami Abu Sneineh (Supervisor),

Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Yaqoub Ashhab (Supervisor),

Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Hashem Tamimi (Internal committee member),

Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Alekos Athanasiadis (External committee member),

Instituto Gulbenkian de Ciencia.

Signature:_____ Date:_____

Thesis Approved

Dr. Murad Abu Sbeih Dean of Graduate Studies and Scientific Research Palestine Polytechnic University

Signature:_____ Date:_____

DECLARATION

I declare that the Master Thesis entitled **"Protein Sequence Data: Laying the Mathematical Foundations to Derive Novel Descriptors"** is my 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 acknowledgement is made in the text.

Berat Huseyin Alisan Kurar

Signature:_____ Date:_____

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 coping or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Berat Huseyin Alisan Kurar

Signature:_____ Date:_____

DEDICATION

To my parents Mrs&Mr Gunes and Huseyin Kurar

To my siblings Namik and Haluk

To my husband Imadudeen

To my kids Fatih and Hamza

ACKNOWLEDGEMENT

Praise be to Allah the Lord of the worlds and may the blessings and peace of Allah be upon the most honored of messengers our master Muhammad and upon all his family and companions.

To my parents, thank you for loving me and being there always. Thank you my mother Gunes the sun, my father Huseyin the earth, for your intellectuality and productivity. I always knew that you believed in me and wanted the best for me.

I am very grateful for my brothers, Namik and Haluk for their constant love and support, no matter how far away I am.

Many thanks to Mrs. Hana and Mr. Ali for the encouragement they have given me through my collegiate carrier. I am honourable to have you as my parents in love.

Thank you my beloved husband Imad, for your continuous encouragement and making me more than I am.

Thank you my dears Fatih and Hamza, for being my amazing blessings.

Finally I would like to acknowledge my academic supervisors Dr. Sami Abu Sneineh and Dr. Yaqoub Ashhab for their assistance and support throughout this process.

Abstract

Proteins' structures and functions play key roles in organisms' life. However, it is timely and costly to experimentally determine the attributes of protein sequences. There is a tremendous growth in the amount of available protein data and the interpretation of information contained in protein sequences is a complex process. Hence it is important to extract features from proteins by descriptors and classify them by computational methods.

Conventional discrete descriptor for proteins is defined as the occurrence frequency of each amino acid type in a protein sequence. This method works well with strong amino acid composition similarities but fails when sequence order across the protein is a strong determinant of the attribute. In addition, conventional sequential descriptors work well when sequence order across the protein is a strong determinant of the attribute but are not applicable to various lengths of proteins. These cases necessitate fixed length descriptors that are able to capture some sequence order effect in protein sequences of various lengths. This thesis proposes a new mathematical definition for the conventional discrete descriptor and new fixed length descriptors with partial sequence order effect as well as new discrete descriptors that considers some similarity among amino acids according to their descriptors.

We performed a comparison with six standard protein descriptors and fourteen novel descriptors; on three classification problems subcellular localization, caspase peptides and DNA binding; by two classification models Random Forest and Support Vector Machines. The experimental results demonstrate the effectiveness of

the new descriptors. Some of the novel descriptors outperform Pseudo Amino Acid Composition in terms of the Area Under Curve and the execution time measurements. Performance differences exist between the descriptors thereby underlining that choosing an appropriate protein descriptor is of paramount protein classification modelling.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Proteins are in the core of an organism's life with their various structure and functions. In-lab methods discover protein sequences faster than their biological attributes. Therefore, computational methods are used to predict the function or structure of proteins based on their sequence information. An important phase in computational methods is extracting informative features from protein sequences [8].

Protein primary representation is the amino acid sequence of the protein [25]. Each amino acid in the sequence is represented by its letter symbol. Protein descriptor is a high dimensional vector of real numbers. A protein descriptor is used to extract features from the primary representation for the machine learning algorithm. Primary representation descriptors are based on substituting amino acids in primary representation with their attributes like sparse values, blosum values, 3D values or physicochemical values, namely amino acid descriptors.

In this thesis we schematize protein descriptors as three types; sequential, hybrid, and discrete. Sequential protein descriptors include full sequence order information. However the length of features varies with the length of protein sequence and is not useful for many classification models. Discrete protein descriptors include no sequence order information. However fixed length of features is useful for the classification models. Hybrid protein descriptors include partial sequence order

information with fixed length of features.

Protein datasets are characterized with various biological properties such as compositional, sequential, structural, evolutional, physicochemical or a hybrid of their subset. Protein descriptors describe proteins with various features such as compositional, sequential, structural, evolutional, physicochemical or a hybrid of their subset. This leads to the hypothesis that a single descriptor might not be optimal for all protein datasets. We aimed at supporting this hypothesis in twospace. Firstly, we proposed a new mathematical definition for conventional discrete descriptor (amino acid composition). This mathematical definition provided us with the flexibility to propose several new discrete and hybrid protein descriptors besides three miscellaneous protein descriptors. Hence we had more descriptors applicable to various lengths of proteins. Secondly, we evaluated the prediction performance of new descriptors and some standard descriptors on three binary classification problems with two classification models Random Forest and Support Vector Machines.

We found that the descriptors proposed in this thesis which comprise the new mathematical definition of conventional discrete descriptor, return good results. In particular they tend to give better prediction performance than the Pseudo Amino Acid Composition. While there seems to be no preferred descriptor that could be utilized for all datasets as prediction results highly dependent on protein datasets, the performance of the protein classification may be enhanced by selecting the optimal protein descriptor. Comparisons among the two classification models showed that different machine learning algorithms with the same descriptor have close performances. Comparisons among the descriptors showed that protein descriptor is a strong determinant of the predictive performance of a classifier. Adequate selection of protein descriptor for a protein dataset is as important as selection of learning algorithm. Bioinformaticians tend to keep using protein descriptor they first encountered, however incorporation of appropriate protein descriptor may help improving the prediction performance.

1.1 Thesis Outline

The remaining chapters of the thesis are organized as follows:

Chapter 2 presents background information for our mathematical model and the two classification models, explains the performance measures that are used.

Chapter 3 defines a mathematical model for conventional discrete descriptor; descriptors raise from this modeling and schematize the defined descriptors.

Chapter 4 describes the datasets used in the experiments.

Chapter 5 reports experimental results on three classification problems and a discussion on the results.

Chapter 6 draws conclusions and proposes some future works.

Chapter 2

Background and Literature Review

This chapter discusses background for this thesis. In the first section we introduce amino acid alphabet, a set of amino acid symbols. We discuss the representation of protein primary structure and substitution matrices for mapping amino acid symbols into high dimensional numeric spaces. Finally we explain supervised learning and two algorithms Random Forest and Support Vector Machines for binary classification and how to evaluate their predictive performances using accuracy and area under curve measures. In the second section we summarize the work and the findings of selected related research in protein sequence based descriptors.

2.1 Background

2.1.1 Amino Acid Alphabet

Proteins are macromolecules consist of different length of chains of amino acids. An amino acid at a particular position in the protein sequence is called a residue. There are 20 naturally occurring amino acids that are abbreviated with one-letter symbols as listed in Table 2.1 [15]. Amino acid alphabet is the finite set of one-letter

Table 2.1: Amino acid names and one-letter symbols

No	Name	Symbol	No	Name	Symbol
1	Alanine	A	11	Leucine	L
2	Arginine	R	12	Lysine	K
3	Asparagine	N	13	Methionine	M
4	AsparticAcid	D	14	Phenylalanine	F
5	Cysteine	C	15	Proline	P
6	Glutamine	Q	16	Serine	S
7	GlutamicAcid	E	17	Threonine	T
8	Glycine	G	18	Tryptophan	W
9	Histidine	H	19	Tyrosine	Y
10	Isoleucine	I	20	Valine	V

symbols.

2.1.2 Protein Primary Representation

A protein over the amino acid alphabet is a finite sequence of one-letter symbols. This finite sequence is the protein primary representation. The length of the protein is the number of residues it is composed of.

2.1.3 Amino Acid Substitution Matrices

Usually computational methods requires numeric values instead of letter symbols. Substitution matrix maps the set of 20 native amino acids into a high dimensional space. Then each amino acid is represented by a numeric vector which is suitable for numerical machine learning algorithms. Each dimension in an amino acid vector scores similarity of that amino acid with others. Dimension values of the vectors can be determined by mathematical or physicochemical attributes of the amino acids. In this thesis we considered sparse, blosum, and 3D substitution matrices.

2.1.3.1 Sparse Matrix

Sparse matrix defines amino acids as 20-bit vectors that are all orthogonal and in equal Euclidean distance to each other (Table 2.2). Each amino acid is only similar

to itself, and represented by 1 and different from all others and represented 0. As a result, amino acids have no similarity in the sparse space. Each amino acid is very specific and is not mutable to another amino acid.

Table 2.2: Sparse matrix

	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

2.1.3.2 BLOSUM Matrix

BLOSUM (Block Substitution Matrix) [14] defines amino acids as 20-bit vectors that are not in equal Euclidean distance to each other (Table 2.3). They are derived by analyzing the mutation frequencies that are seen in the block sequence alignments of the known protein families [14]. There are different BLOSUM matrices based on different cutoffs in the sequence identity within a block. We used BLOSUM62 matrix derived from the alignments in which proteins are 62% identical. Most identical amino acids are scored most positive and least identical amino acids are scored the least negative. As a result, amino acids have some similarity in the blosum space. Each dimension in an amino acid vector corresponds to mutability rate of that amino acid to others.

Table 2.3: BLOSUM62 matrix

	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

2.1.3.3 3D Matrix

3D (Three Dimensional) matrix [20] defines amino acids as 3-bit vectors that are not in equal Euclidean distance to each other (Table 2.4). They are derived by principal component analysis (PCA) of the BLOSUM62 matrix. As a result, amino acids are linearly uncorrelated vectors that best explain BLOSUM62 data in the three dimensional space [20].

2.1.4 Protein Descriptors for Protein Primary Representation

Protein descriptors are methods to extract features from protein primary representation. That means descriptors make protein primary representation usable by numeric machine learning algorithms. This thesis studied two basic descriptors: conventional discrete descriptor and conventional sequential descriptor.

Table 2.4: 3D matrix

	1	2	3
A	0.189	-3.989	1.989
R	5.007	0.834	-2.709
N	7.616	0.943	0.101
D	7.781	0.030	1.821
C	-5.929	-4.837	6.206
Q	5.480	1.293	-3.091
E	7.444	1.005	-2.121
G	4.096	0.772	7.120
H	3.488	6.754	-2.703
I	-7.883	-4.900	-2.230
L	-7.582	-3.724	-2.740
K	5.665	-0.166	-2.643
M	-5.200	-2.547	-3.561
F	-8.681	4.397	-0.732
P	4.281	-2.932	2.319
S	4.201	-1.948	1.453
T	0.774	-3.192	0.666
W	-8.492	9.958	4.874
Y	-6.147	7.590	-2.065
Y	-6.108	-5.341	-1.953

2.1.4.1 Conventional Discrete Descriptor (Amino Acid Composition)

In the literature Conventional Discrete descriptor (CD) is usually referred to by Amino Acid Composition (AAC). It is a vector of 20 dimensions that includes the occurrence number of 20 amino acids in a protein [25].

2.1.4.2 Conventional Sequential Descriptor (Orthonormal Encoding)

In the literature Conventional Sequential descriptor (CS) is usually referred to by Orthonormal Encoding (OE). It is a vector of $20xn$ dimensions that includes the concatenation of sparse vectors of amino acids in a protein of n residues [24].

2.1.4.3 Pseudo Amino Acid Composition

Pseudo Amino Acid Composition (PAAC) [6] is also called type 1 pseudo amino acid composition. It is a vector of $(20 + \lambda)$ dimensions and includes information of the amino acid composition as well as the information of sequence order effect.

Definition 2.1.1. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Pseudo Amino*

Acid Composition of protein P is

$$PAAC\left(P\right)=\left[x_1,x_2,\ldots x_{20+1},\ldots x_{20+\lambda}\right]$$

where

$$x_u=\begin{cases}\frac{f_u}{\sum_{i=1}^{20}f_i+\omega\sum_{j=1}^{\lambda}\theta_j},\ (1\leq u\leq 20)\\ \frac{\omega\theta_{u-20}}{\sum_{i=1}^{20}f_i+\omega\sum_{j=1}^{\lambda}\theta_j},\ (20+1\leq u\leq 20+\lambda)\end{cases}$$

[6]

f_i is the normalized occurrence frequency of 20 native amino acids in protein P, ω is the weight factor for sequence order effect, and θ_i is the j-tier sequence correlation factor for the protein given by the following.

$$\begin{cases}\theta_1=\frac{1}{n-1}\sum_{i=1}^{n-1}\Theta\left(p_i,p_{i+1}\right)\\ \theta_2=\frac{1}{n-2}\sum_{i=1}^{n-2}\Theta(p_i,p_{i+2}),\ (\lambda<n)\\ \ldots\\ \theta_\lambda=\frac{1}{n-\lambda}\sum_{i=1}^{n-\lambda}\Theta(p_i,p_{i+\lambda})\end{cases}$$

θ_1 is called the first-tier correlation factor that reflects the sequence order correlation between all the most adjacent residues along the protein (Figure 2.1a), θ_2 is called the second-tier correlation factor that reflects the sequence order correlation between all the second most adjacent residues along the protein (Figure 2.1b), θ_3 is called the second-tier correlation factor that reflects the sequence order correlation between all the third most adjacent residues along the protein (Figure 2.1c). Correlation function Θ is given by the following.

$$\Theta\left(p_i,p_j\right)=\frac{1}{3}\{\left[H_1\left(p_j\right)-H_1\left(p_i\right)\right]^2+\left[H_2\left(p_j\right)-H_2\left(p_i\right)\right]^2+\left[M\left(p_j\right)-M\left(p_i\right)\right]^2\}$$

Where $H_1\left(p_i\right)$, $H_2\left(p_i\right)$ and $M\left(p_i\right)$ are respectively, the hydrophobicity value, hydrophilicity value, and side-chain mass of the amino acid p_i and they are all con-

verted by the following equation:

$$\begin{cases} H_1(i) = \dfrac{H_1{}^o(i) - \sum_{i=1}^{20} \frac{H_1{}^o(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20}\left[H_1{}^o(i) - \sum_{i=1}^{20} \frac{H_1{}^o(i)}{20} \right]}{20}}} \\ H_2(i) = \dfrac{H_2{}^o(i) - \sum_{i=1}^{20} \frac{H_2{}^o(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20}\left[H_2{}^o(i) - \sum_{i=1}^{20} \frac{H_2{}^o(i)}{20} \right]}{20}}} \\ M(i) = \dfrac{M^o(i) - \sum_{i=1}^{20} \frac{M^o(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20}\left[M^o(i) - \sum_{i=1}^{20} \frac{M^o(i)}{20} \right]}{20}}} \end{cases}$$

where $H_1{}^o(i)$ is the original hydrophobicity value, $H_2{}^o(i)$ the original hydrophilicity value, and $M^o(i)$ the original side-chain mass of the i^{th} amino acid in the ordered list O of amino acid one-letter symbols.

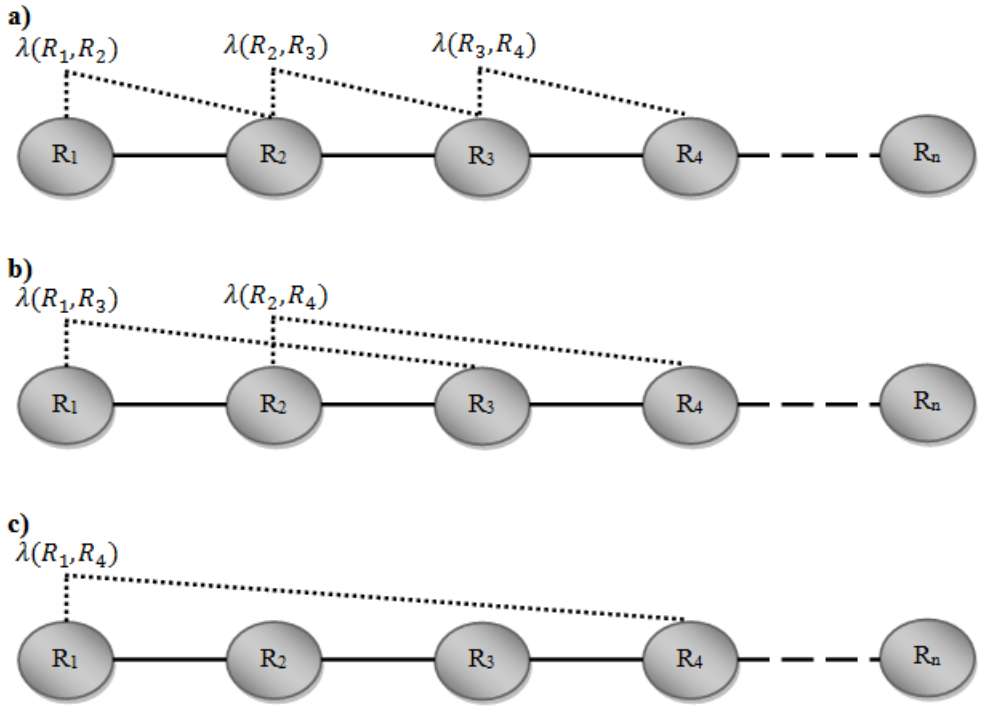

Figure 2.1: a) The first-tier, b) the second-tier, and c) the third-tier sequence order correlation mode along a protein sequence.

2.1.5 Supervised Learning

Given a training set, supervised learning maps a function $h : \mathbf{X} \to Y$ such that $h(\mathbf{x})$ predicts the corresponding value of y.

$\mathbf{X}$ is the space of input values, and Y is the space of output values. Input values are referred as feature vectors and denoted by $\mathbf{x}_i$. Output values are referred as labels and denoted by y_i. A pair $(\mathbf{x}_i, y_i)$ is a called a training example and a list of m training examples $\{(\mathbf{x}_i, y_i)\,; i = 1, 2, \ldots m\}$ is called a training set.

Usually $\mathbf{x} \subseteq R^n$ while $y \in \{-1, +1\}$ for binary classification which is the case in this thesis.

2.1.6 Random Forests

Random Forests (RF) [4] is a collection of decision trees. Each tree is trained on a random sample of dataset with replacement and randomly selected feature vector elements at its each node. For any new input vector each tree votes a class then the class that has the most votes is predicted.

Definition 2.1.2. *A random forest is a classifier consisting of a collection of tree structured classifiers* $\{h(\boldsymbol{x}, \boldsymbol{\theta}_k),\ k = 1, 2, \ldots\}$ *where the* $\{\boldsymbol{\theta}_k\}$ *are independent identically distributed (iid) random vectors and each tree casts a unit vote for the most popular class at input* $\boldsymbol{x}$ *[4].*

The trees in RF are CART [5] decision trees besides that bagging and random feature selection is used in tandem [4].

Random vectors $\boldsymbol{\theta}$ are generated to govern the growth of each decision tree in the collection. $\boldsymbol{\theta}_k = (\theta_{k1}, \theta_{k2}, \ldots \theta_{kp})$ defines the parameters of the tree for the classifier $h_k(\mathbf{x})$. Since $\boldsymbol{\theta}$ are iid then $\boldsymbol{\theta}_k$ is independent of the previous random vectors $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots \boldsymbol{\theta}_{k-1}$ but has the same distribution with them.

For example in bagging $\boldsymbol{\theta}_k$ determines bootstrap training set T_k for the tree $h_k(\mathbf{x})$ from the full training set $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_m, y_m)\}$. In random feature selection $\boldsymbol{\theta}_k$ determines feature subvector $\mathbf{x}_k$ for the tree $h_k(\mathbf{x})$ from the full feature vector $\mathbf{x} = (x_1, x_2, \ldots x_n)$.

In classification problems each tree generates its own prediction for any input vector. All the predictions for an input vector are combined by voting (Figure 2.2).

RF can also calculate the class probability of an input vector based on votes for each class.

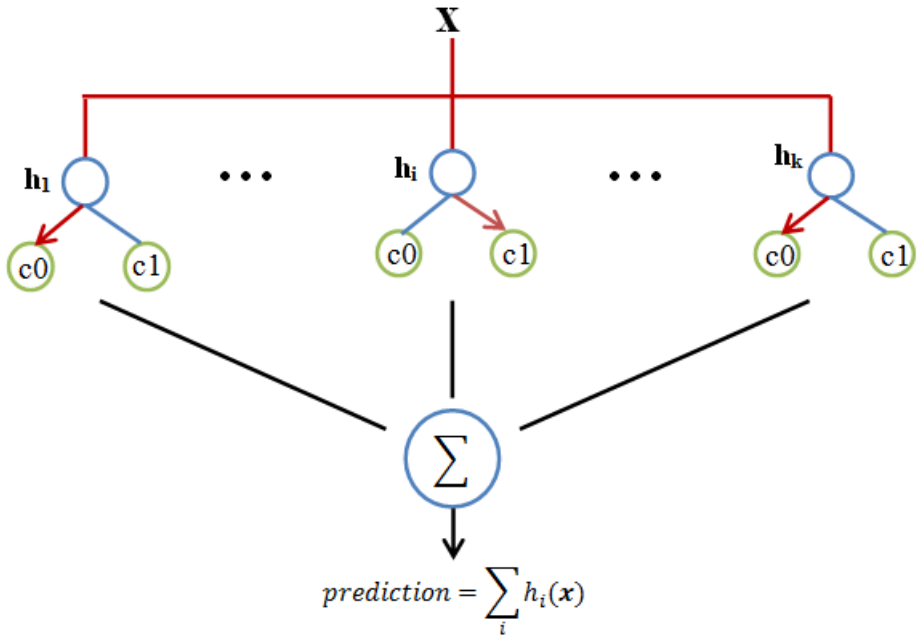

Figure 2.2: Predictions for input vector **x** is combined by voting

2.1.7 Support Vector Machines

Support Vector Machines [1] (SVM) [10] for binary classification problem first maps input values to a linearly separable high dimensional space then draws an optimal separating hyperplane between the positive and negative examples. Optimal separating hyperplane separates the examples with the maximum margin possible among them which optimizes the generalization and prevents overfitting.

Let R^n denote the original input space, R^N denote the new input space and ϕ denotes the mapping (Figure 2.3) then

$$\phi : R^n \rightarrow R^N$$

[1] In this thesis the term SVM refers to classification with support vector methods.

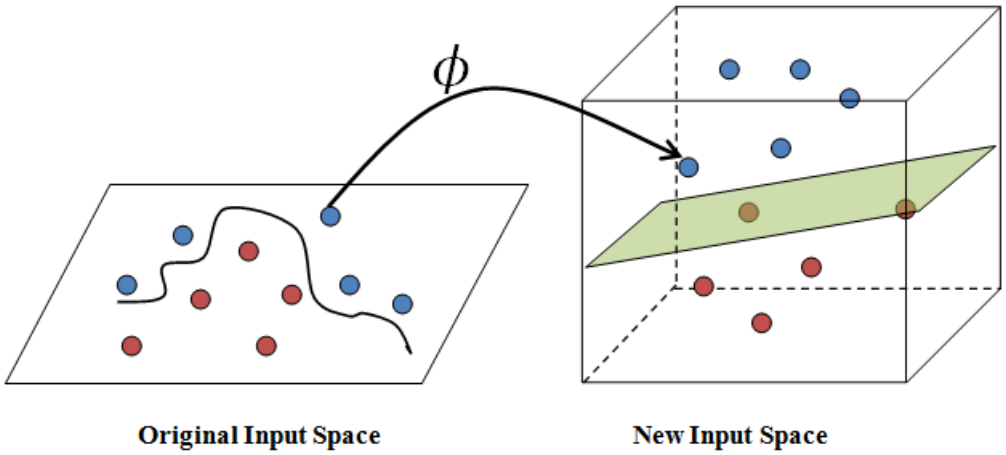

Figure 2.3: ϕ maps input values to a linearly separable high dimensional space

Thus the training example $(\mathbf{x}_i, y_i)$ becomes $(\phi(\mathbf{x}_i), y_i)$ where $\phi(\mathbf{x}_i) \in R^N$ and $y_i \in \{-1, +1\}$.

In this work we used SVM with Radial Basis Function (RBF) [29] kernel. RBF kernel is defined as

$$K(\mathbf{x}_1, \mathbf{x}_2) = exp(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{2\sigma^2})$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are feature vectors and σ is a parameter for the width of the kernel.

Suppose there is a hyperplane in R^N that separates the positive examples from negative examples. It is referred as a separating hyperplane H and its equation is

$$H : \boldsymbol{\omega}.\phi(\mathbf{x}) + b = 0$$

$\boldsymbol{\omega}$ is normal vector to the hyperplane and $\frac{|b|}{||\boldsymbol{\omega}||}$ is the distance from hyperplane to the origin.

Since each example is at least β away from the hyperplane for some $\beta > 0$ then we can normalize the hyperplane equation such that training examples satisfy the following:

$$\boldsymbol{\omega}.\phi(\mathbf{x}_i) + b \geq +1 \; if \; y_i = +1$$

$$\boldsymbol{\omega}.\phi(\mathbf{x}_i) + b \leq -1 \; if \; y_i = -1$$

which is equivalently

$$y_i(\boldsymbol{\omega}.\phi(\mathbf{x}_i) + b) - 1 \geq 0 \; for \; i = 1, 2, 3, \ldots m$$

The hyperplane $H_1 : \boldsymbol{\omega}.\phi(\mathbf{x}) + b = +1$ is with distance from the origin $\frac{|+1-b|}{||\boldsymbol{\omega}||}$.
The hyperplane $H_2 : \boldsymbol{\omega}.\phi(\mathbf{x}) + b = -1$ is with distance from the origin $\frac{|-1-b|}{||\boldsymbol{\omega}||}$.
Then the margin of the separating hyperplane is $\frac{2}{||\boldsymbol{\omega}||}$ (Figure 2.4).

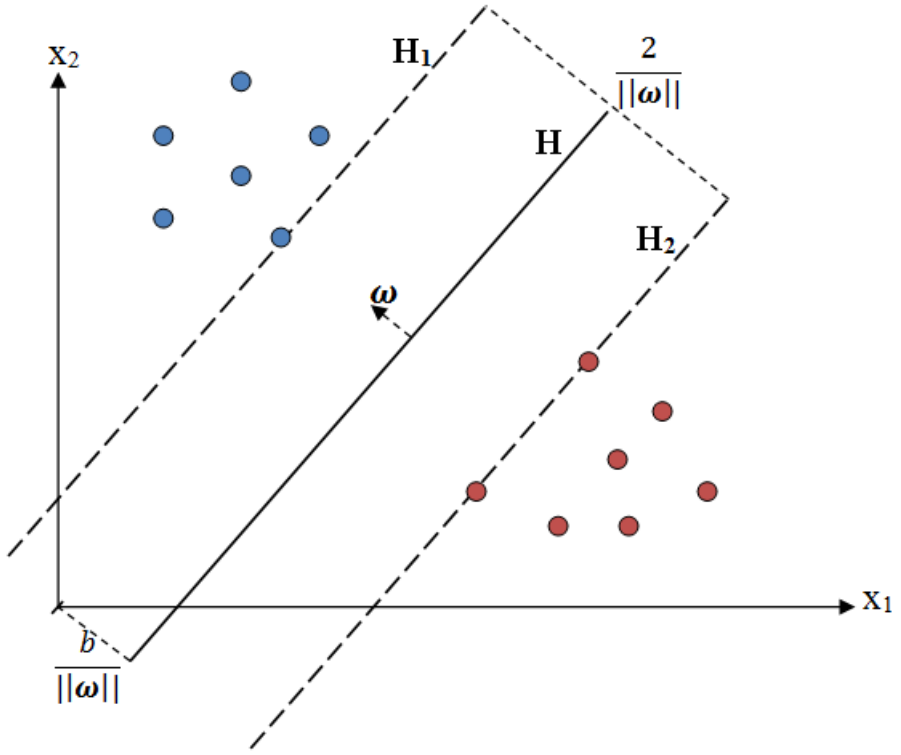

Figure 2.4: Margin of the separating hyperplane

In general training set may not be separable with a hyperplane. That's why slack variables ξ_i are introduced with $\xi_i \geq 0$ for $i = 1, 2, \ldots m$. Then training examples satisfy the following:

$$\boldsymbol{\omega}.\phi(\mathbf{x}_i) + b \geq +1 - \xi_i \quad if \; y_i = +1$$

$$\boldsymbol{\omega}.\phi\left(\mathbf{x}_i\right)+b\leq\ -1+\xi_i\quad if\ y_i=-1$$

which is equivalently

$$y_i\left(\boldsymbol{\omega}.\phi\left(\mathbf{x}_i\right)+b\right)-1+\xi_i\geq 0\ for\ i=1,2,3,\ldots m$$

$$\xi_i\geq 0\ for\ \xi_i=1,2,3,\ldots m$$

Since SVM's goal is to separate the training examples with largest margin possible among them then it must solve the following optimization problem with $C>0$ is a constant to set the relative importance of slack variables.

$$minimize\ \ \left|\left|\boldsymbol{\omega}\right|\right|^2+C\sum_i\xi_i$$

$$subject\ to\ \ y_i\left(\boldsymbol{\omega}.\phi\left(\mathbf{x}_i\right)+b\right)-1+\xi_i\geq 0\ for\ i=1,2,3,\ldots m$$

$$\xi_i\geq 0\ for\ \xi_i=1,2,3,\ldots m$$

Using Lagrange multipliers this problem is reformulated in the following way

$$maximize\ \sum_i\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j(\phi\left(\mathbf{x}_i\right).\phi\left(\mathbf{x}_j\right))$$

$$subject\ to\ \sum_i\alpha_i y_i=0\ and\ 0\leq\alpha_i\leq C\ for\ i=1,2,3,\ldots m$$

Through solving the reformulated problem, values for the α_i, and the ξ_i, are obtained and the following equations are satisfied

$$\boldsymbol{\omega}=\sum_i\alpha_i y_i\phi\left(\mathbf{x}_i\right)\tag{2.1}$$

$$\alpha_i(y_i\left(\boldsymbol{\omega}.\phi\left(\mathbf{x}_i\right)+b\right)-1)=0\ \forall i\tag{2.2}$$

$$y_i\left(\boldsymbol{\omega}.\phi\left(\mathbf{x}_i\right)+b\right)-1\geq 0\ \forall i\tag{2.3}$$

Thus $\boldsymbol{\omega}$ can be obtained from equation 2.1 and b can be obtained from equation

Table 2.5: Confusion matrix

		True	
		Label +1	Label -1
Predicted	Label +1	tp	fp
	Label -1	fn	tn

2.2 as the average of b values for each i.

Solving this optimization problem and obtaining the maximum margin separating hyperplane is the training of an SVM. For classification of an input vector $\mathbf{x}$, SVM checks on which side of the hyperplane it falls.

2.1.8 Confusion Matrix

Let $y_{pred} = \{\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m\}$ be a collection of predicted labels and $y_{true} = \{y_1, y_2, \ldots y_m\}$ be the collection of their true labels where $y_i, \bar{y}_i \in \{-1, +1\}$. Then the evaluation of a classifier is to compare the labels in y_{pred} with the corresponding labels in y_{true}.

A confusion matrix (Table 2.5) is a contingency table which includes counts of True Positives (tp), True Negatives (tn), False Positives (fp) and False Negatives (fn) [3] where

$$tp = card(\bar{y}_i = 1 \; and \; y_i = 1)$$

$$tn = card(\bar{y}_i = -1 \; and \; y_i = -1)$$

$$fp = card(\bar{y}_i = 1 \; and \; y_i = -1)$$

$$fn = card(\bar{y}_i = -1 \; and \; y_i = 1)$$

2.1.9 Accuracy

Accuracy (ACC) is a measure of comparing predictions of a classifier against their true labels. It is the ratio of number of correct predictions and the number of total

predictions [3].

$$Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ predictions} = \frac{tp + tn}{tp + tn + fp + fn}$$

Accuracy measure is easily computable but if two class sizes are not equal accuracy can be skewed. For example let 80 examples have label +1 and 5 examples have label -1. A classifier that predicts all 100 examples have label +1, would have an accuracy of 80% trivially. However it can be used to get a quick indication of how well a classifier is.

2.1.10 Area under the ROC Curve

The area under the ROC curve (AUC) [22] is another measure to evaluate a classifier. It is based on the variation of a classifier parameter. We consider the class membership probability as the varying classifier parameter. Class membership probability is the measure to the degree of which an input value belongs to a class.

Let $(\mathbf{x}_i, y_i)$ is the i^{th} example of the training set and s_i is the score for $\mathbf{x}_i$'s class membership probability $if\ s_i > T\ then\ y_i = +1\ else - 1$, where T is a threshold value and $0 < T < 1$.

Let a classifier depends on a parameter $\alpha \in [a, b] \subseteq R$. During classification let α varies from a to b in a finite number of increments ω.

$$a = \alpha_1 < \alpha_2 < \alpha_3, < \ldots \alpha_{\omega-1} < \alpha_\omega = b$$

For each value $\alpha \in \{\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_\omega\}$ classifier will output the collection

$$y_{pred,\alpha} = \{\bar{y}_{1,\alpha}, \bar{y}_{2,\alpha}, \ldots \bar{y}_{m,\alpha}\}$$

So a confusion matrix for each value of α can be calculated (Table 2.6).

Let tpr_α and fpr_α be the true positive rate and false positive rate for the

Table 2.6: Confusion matrix for each value of α

		True	
		Label +1	Label -1
Predicted	Label +1	tp_α	fp_α
	Label -1	fn_α	tn_α

classifier with parameter α, respectively. They are given by

$$tpr_\alpha = \frac{tp_\alpha}{tp_\alpha + fn_\alpha}$$

$$fpr_\alpha = \frac{fp_\alpha}{fp_\alpha + tn_\alpha}$$

Points of (tpr_α, fpr_α) can be plotted in R^2 for $\alpha \in \{\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_\omega\}$. ROC curve is the curve that joins these points (Figure 2.5). AUC is the integral of this curve [12] and given by

$$AUC = \left| \sum_{i=1}^{\omega-1} (fpr_{\alpha_i} - fpr_{\alpha_{i+1}}) tpr_{\alpha_{i+1}} \right|$$

2.1.11 K-Fold Cross Validation

K-fold cross validation [16] is a method of dividing training set to determine optimal classifier parameters and evaluate the classifier. In k-fold cross validation training set is randomly divided into k mutually exclusive subsets $S_{L1}, S_{L2}, S_{L3}, \ldots S_{Lk}$ of approximately equal size. Each subset is called a fold. In stratified cross validation each fold contains approximately same proportions of labels as the training set.

The classifier is trained and tested k times. Each time it is trained on $S_L - S_{Lf}$, tested on S_f where $f \in \{1, 2, 3, \ldots k\}$. For each time a performance score s_f is assigned.

The mean $\bar{s}$ of the scores $s_1, s_2, s_3, \ldots s_k$

$$\bar{s} = \frac{\sum_{f=1}^{k} s_i}{k}$$

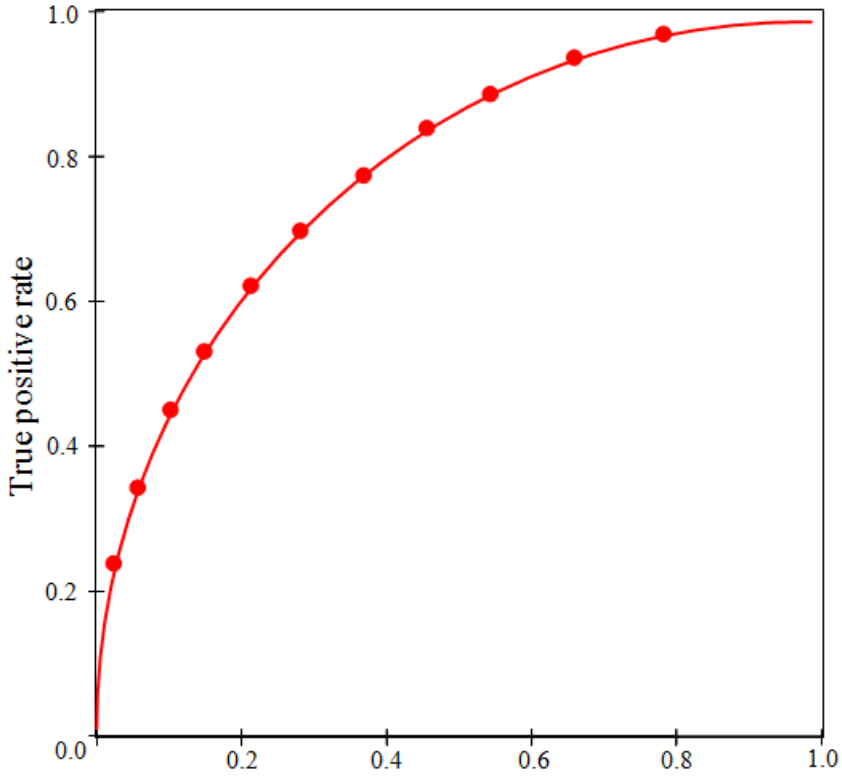

Figure 2.5: A ROC curve where α increased from a to b in $\omega = 11$ increments. The points of (tpr_α, fpr_α) are plotted in R^2 and the curve joining these points is the ROC curve.

is considered as an estimate of classifier's performance.

K-fold cross validation guarantee that each training example is trained $k-1$ times and tested once. Therefore it reduces variance while increasing the bias. However stratified cross validation gives better scheme in variance and bias [16], therefore used in our work.

2.2 Literature Review

Many machine learning methods are used to predict protein attributes. Numerous methods proposed to extract useful information from protein sequences. In this thesis we employed machine learning to compare sequence based protein descriptors in their ability to contain useful information. In this section we summarize the research related to sequence based protein descriptors used to extract useful information from proteins.

Orthonormal encoding is frequently used to represent protein sequences [24]. As an early example, (Qian et al. 1988) used orthonormal encoding and neural networks to construct a predictive model for the secondary structure of globular proteins. They concluded that no method based solely on protein sequence could produce significantly better results. As a recent example, (Nanni et al. 2011) performed an extensive comparison with several encoding schemes including orthonormal encoding on three classification problems. They addressed the issue of which encoding method promises the best results for classification problems. They also concluded that frequently used orthonormal encoding is inferior compared to other methods. Among the descriptors that we studied, Sequential Sparse encoding is the same as orthonormal encoding. Sequential Discrete Sparse and Discrete Sequential Sparse descriptors are also based on sparse matrix as well as the orthonormal encoding. However Sequential Discrete Sparse and Discrete Sequential Sparse descriptors have fixed length and are able to represent compositional character of proteins partially.

Orthonormal encoding can lead to a complex model and overfitting due to high number of dimensions per residue. Reduced encoding schemes use less number of dimensions to describe an amino acid thus leads to less complex models to describe the same problem. In an early research in protein attribute prediction; (Skolnick et al. 1997) defined 10-bit binary code for each amino acid. Binary codes were based on the amino acid features described by (Taylor and Ramsay 1986). At each bit 1 indicates that the corresponding feature is present in the amino acid and 0 indicates that the corresponding feature absent in the amino acid. Therefore, amino acids that share common features have similar codes. They trained an Artificial Neural Network on a MHC-I binding dataset and reported their results as highly accurate predictions. All our discrete and hybrid descriptors with their fixed length of dimensions, lead to simpler models just like the reduced encoding schemes.

Blosum [14] encoding is also frequently used in researches [11]. (Nielsen et al. 2003) applied Blosum50 encoding for T-cell epitope prediction. They showed that the usage of Blosum encoding leads to an increased performance over the orthogonal encoding. Blosum encoding describes some similarity and dissimilarity among the amino acids so this helps neural network to generalize better than with orthogonal encoding in which all amino acids are equally similar to each other. Recently (Menor et al. 2012) built a model to predict protein phosphorylation sites using blosum encoding and amino acid composition. They exposed that blosum encoding is also superior to amino acid composition in terms of predictive performance. Among the descriptors studied in this thesis, Discrete Blosum corresponds to this method. However Discrete Blosum differentiates from this method by fixed number of dimensions for any protein length. Sequential Discrete Blosum and Discrete Sequential Blosum descriptors differentiate from this method by fixed number of dimensions as well as the ability to represent some sequence order effect.

(Nakashima et al. 1986) examined the ability of amino acid composition in discriminating protein folding types. They expressed amino acid composition of a protein as a point in a 20 dimensional space. Their results show that amino acid

compositions of five folding types are clustered in different regions of the space. They used distance metric to classify new proteins into the five folding types and achieved a moderate accuracy.

Lately many methods for predicting protein attributes were based on amino acid composition. (Chou et al. 1999) proposed a covariant discriminant algorithm to predict the subcellular location of a protein according to its amino acid composition. (Feng et al. 2005) introduced a boosting classifier that uses amino acid composition to represent the sample of proteins domains. (Kumar et al. 2015) developed classification models using amino acid composition for predicting antihypersensitive peptides. Amino acid composition is the descriptor that we state a mathematical definition for and named as Discrete Sparse descriptor.

The main limitation of the amino acid composition is the missing sequence order effect. To deal with this difficulty, (Chou and Kuo-Chen 2001) introduced pseudo amino acid composition (PAAC). They reported a remarkable improvement in prediction quality of protein cellular attributes. PAAC has been widely used in various protein classification problems. (Qiu et al. 2010, Cai et al. 2005, Zhou et al. 2007) used PAAC to predict the enzymatic attribute of proteins. (Zhang et al. 2008, Gao et al. 2005) used PAAC to predict protein subcellular location. (Nanni and Lumini 2008) applied genetic algorithm to find features of PAAC for submitochondria localization. (Kumar et al. 2010) applied the same method of (Nanni and Lumini 2008) for prediction of apoptosis protein locations. In our descriptors scheme (Figure 3.1) sequential descriptors have full sequence order effect whereas hybrid descriptors have partial sequence order effect like PAAC.

(Guda et al.2004) analyzed the 25 residues of the N terminus, 25 residues of the C terminus and the remaining residues in mitochondrial proteins. They observed that amino acid composition of N terminal residues and remaining residues was different. Although C terminal residues have different amino acid composition than the remaining residues, it was not as significant as the N terminal residues. Inspiring by this biological rationale (Kumar et al. 2006) introduced split amino

acid composition (SAAC). SAAC represents protein sequences by a fixed length of 60 dimensions. 20 dimensions is for amino acid composition of 25 amino acids of N terminus, 20 dimensions is for amino acid composition of 25 amino acids of C terminus and 20 dimensions is for amino acid composition of remaining residues. They stated that the advantage of SAAC over amino acid composition is that it provides a specific weight for the signals at the N or C terminus. All our Discrete Sequential descriptors also split a protein sequence and then encode each split. However the rationale behind this splitting has no biological meaning but mathematical. Consequently number of splits can be increased as much as the prediction performance increases. Splits are not necessarily for N and C terminus, but in equal number of residues among the protein sequence.

Chapter 3

Methodology

3.1 Mathematical Definitions for Conventional Concepts

Proteins have four levels of representation, primary, secondary, tertiary and quaternary [21]. In this thesis we focus on protein primary representation, which is the sequence of amino acid residues. Primary representation can serve as an indicator of protein attributes to be predicted. For this aim, amino acid composition and orthonormal encoding is widely used to encode the protein primary representation. These concepts are well known in the literature but void of standard mathematical definitions. In the following, we make our own mathematical definitions for these basic concepts in order to model our descriptors consistently.

3.1.1 Amino Acid Alphabet

Amino acid alphabet is described in chapter 2. Based on that description we can make the following definition for the amino acid alphabet.

Definition 3.1.1 (Amino Acid Alphabet)**.** *Let set A be the amino acid alphabet*

then

$$A = \{a_1, a_2, ...a_{20}\}$$

where a_i *is the* i^{th} *element in the ordered list* O *of amino acid one-letter symbols*

$$O = (A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V)$$

3.1.2 Protein Primary Representation

Definition 3.1.1 means that each amino acid is represented by a distinct letter. Then protein primary representation (chapter 2) is a sequence of letters as defined in the following.

Definition 3.1.2 (Protein Primary Representation)**.** *Let* P *be the primary representation of a protein then*

$$P = (p_1, p_2, p_3, \ldots p_n)$$

where $p_i \in A$ *for* $i = 1, 2, 3, \ldots n$ *and* n *is the protein length.*

3.1.3 Amino Acid Descriptors

Amino acids are represented by letters and proteins are represented by sequence of letters. However letters are not quantitative values that are comparable among each other which is a necessity for most of the machine learning algorithms. Therefore amino acids are encoded into numerical vectors. In this thesis we consider three types of amino acid encodings; sparse, blosum and 3D vectors. Based on the description of sparse, blosum and 3D matrices in chapter 2 we define sparse, blosum and 3D vectors as in the following.

Definition 3.1.3 (Sparse Vector)**.** *Let* $S(a_i)$ *be the sparse vector of amino acid* a_i *then*

$$S\left(a_i\right) = \boldsymbol{s}_i = (v_{i1}, v_{i2}, v_{i3}, \ldots v_{i20})$$

where v_{ij} are elements of sparse matrix, for $i = 1, 2, 3, \ldots 20$ and $j = 1, 2, 3, \ldots 20$

Definition 3.1.4 (Blosum Vector)**.** *Let $B(a_i)$be the blosum vector of amino acid a_i then*

$$B\left(a_i\right) = \boldsymbol{b}_i = \left(v_{i1}, v_{i2}, v_{i3}, \ldots v_{i20}\right)$$

where v_{ij} are elements of BLOSUM62 matrix, for $i = 1, 2, 3, \ldots 20$ and $j = 1, 2, 3, \ldots 20$

Definition 3.1.5 (3D Vector)**.** *Let $T(a_i)$ be the 3D vector of amino acid a_i then*

$$T\left(a_i\right) = \boldsymbol{t}_i = \left(v_{i1}, v_{i2}, v_{i3}\right)$$

where v_{ij} are elements of 3D matrix, for $i = 1, 2, 3, \ldots 20$ and $j = 1, 2, 3$

3.1.4 Conventional Descriptors

Using the above definitions we make the following consistent definitions for widely used amino acid composition and orthonormal encoding. In order to have a meaningful and consistent naming convention with our modeling, we renamed them as Conventional Discrete descriptor and Conventional Sequential descriptor correspondingly.

Definition 3.1.6 (Conventional Discrete Descriptor)**.** *Given the protein $P = (p_1, p_2, p_3, \ldots p_n)$ then Conventional Discrete descriptor of protein P is*

$$CD\left(P\right) = \left(C(a_1), C(a_2), \ldots C(a_{20})\right)$$

where $C\left(a_i\right)$ is the number of occurrences of amino acid a_i in protein P.

Definition 3.1.7 (Conventional Sequential Descriptor)**.** *Given the protein $P = (p_1, p_2, p_3, \ldots p_n)$ then Conventional Sequential descriptor of protein P is*

$$CS\left(P\right) = \left(S\left(p_1\right) || S\left(p_2\right) || \ldots S\left(p_n\right)\right)$$

3.2 A Mathematical Model for Conventional Discrete Descriptor

Traditionally discrete descriptor is defined as the occurrence frequency of each amino acid in a protein sequence and referred as amino acid composition. This method involves no sequence order effect of the amino acids in the sequence. In the other extreme, sequential descriptor involves full sequence order effect. Protein datasets have varying features to be considered when needed to be classified. Some may need to be discriminated by their amino acid composition, and some by the order of its amino acids and some by a hybrid of both.

Due to these requirements we state a mathematical definition (Definition 3.2.1) for Conventional Discrete descriptor that parameterizes amino acids' contribution into the descriptor. Then we used this parameterization to design new descriptors that consider amino acid mutability, or 3-D amino acid attributes as well as sequence order effect or amino acid composition or both. We schematize the descriptors in Figure 3.1 indicating proposed ones in bold.

3.2.1 Descriptor 1: Discrete Sparse (DS)

Definition 3.2.1. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then the Discrete Sparse descriptor of* P *is*

$$DS(P) = \sum_{i=1}^{n} S(p_i)$$

According to our definition, Discrete Sparse descriptor is the summation of sparse vectors of amino acids that constitute the protein. Following theorem and its proof shows that our definition for Discrete Sparse descriptor leads to same output as the Conventional Discrete descriptor.

Theorem 3.2.1. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then*

$DS(P) = CDD(P)$.

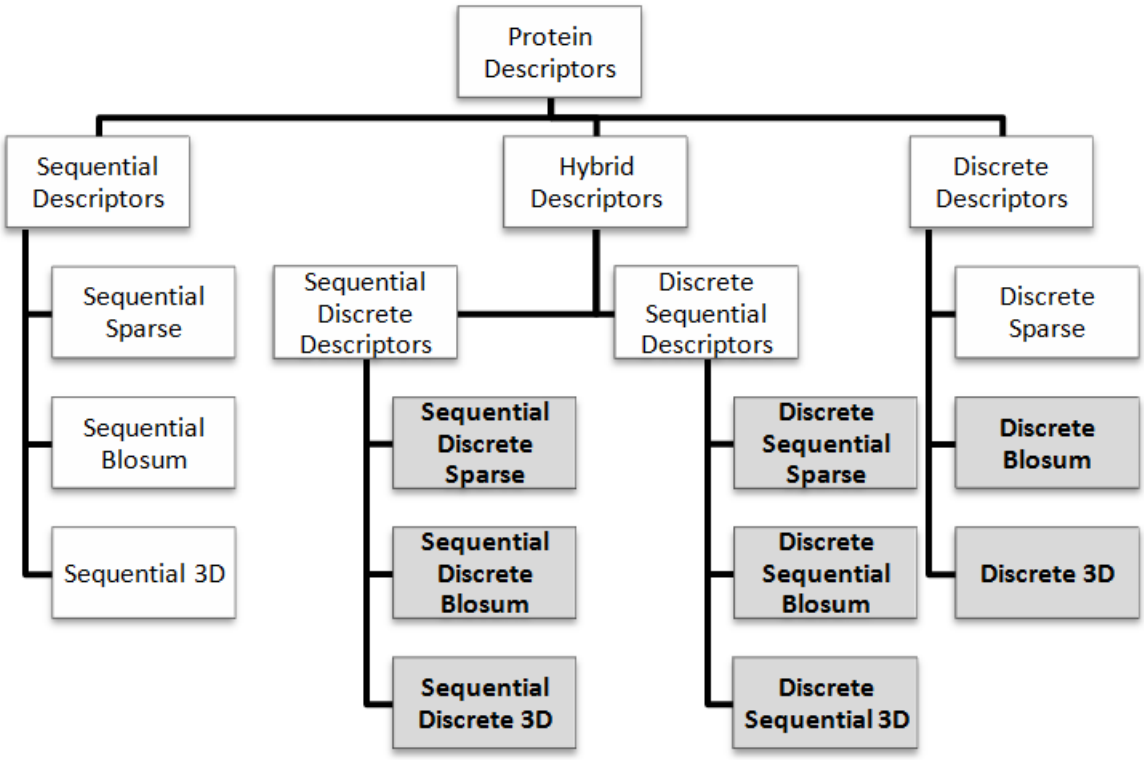

Figure 3.1: A scheme of protein descriptors. Novel descriptor groups are highlighted in grey.

Proof:

$$DS(P) = \sum_{i=1}^{n} S(p_i) = \sum_{i=1}^{20} C(a_i)\, \boldsymbol{s}_i = (C(a_1), C(a_2), \ldots C(a_{20})) = CDD(P)$$

3.3 Descriptors Based on Mathematical Modeling of Discrete Representation

3.3.1 Discrete Descriptors

As we mentioned above, our model parameterizes amino acids' contribution into the descriptor. The parameterization is exactly on the dimension values of the amino acid vectors. We proved that Discrete Sparse descriptor is equal to the Conventional Discrete descriptor. Likewise Discrete Sparse descriptor is the summation of sparse

vectors of amino acids in the protein. This implies that in this summation, to encode the protein sequence we can use blosum or 3D vectors as well. According to our naming convention we named these descriptors as Discrete Blosum and Discrete 3D correspondingly.

3.3.1.1 Descriptor 2: Discrete Blosum (DB)

Definition 3.3.1. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Discrete Blosum descriptor of protein* P *is*

$$DB(P) = \sum_{i=1}^{n} B(p_i)$$

3.3.1.2 Descriptor 3: Discrete 3D (DT)

Definition 3.3.2. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Discrete 3D descriptor of protein* P *is*

$$DT(P) = \sum_{i=1}^{n} T(p_i)$$

3.3.1.3 Properties of Discrete Descriptors

In discrete descriptors;

- if used sparse vectors then amino acids are all orthogonal and in equal distance to each other. Therefore each amino acid is only similar to itself.

- if used blosum vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to mutability rate.

- if used 3D vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to their 3D attributes.

Besides contributing amino acids' mutability and 3D attributes into the descriptors, Discrete Blosum and Discrete 3D descriptors have no sequence order effect just as the Conventional Discrete descriptor. They are applicable to proteins with

various lengths. Their output has a dimensions where a is the number dimensions of amino acid vectors.

3.3.1.4 Concretion of Discrete Descriptors on Figure

Discrete descriptors can be concreted in Figure 3.2 assuming the followings:

Each amino acid, an a dimensional vector, is represented as a ring notated in its center with number of dimensions.

A protein of n residues is represented as a sequence of n amino acid rings.

A discrete descriptor, a d dimensional vector, of a protein is represented as a ring notated in its center with number of dimensions.

Since a discrete descriptor is sum of vectors of amino acids in the protein then $d = a$ regardless of protein length n.

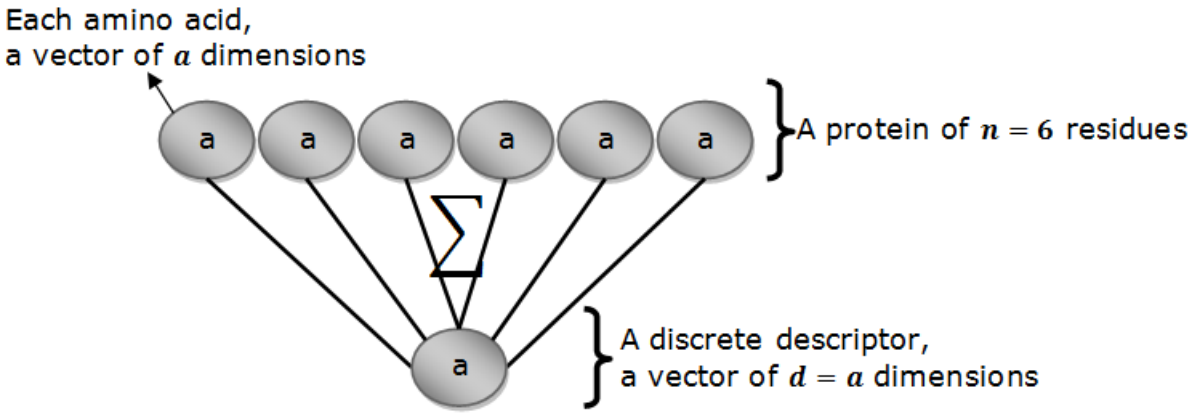

Figure 3.2: A discrete descriptor

3.3.2 Sequential Descriptors

As we schematize in Figure 3.1, in opposite side of the discrete descriptors with no sequence order effect, there are the sequential descriptors with full sequence order effect. In sequential descriptors the parameterization is also on the dimension values of the amino acid vectors. We define that Sequential Sparse descriptor is the concatenation of sparse vectors of amino acids in the protein just like the

Conventional Sequential descriptor. This implies that in this concatenation to encode the protein sequence, we can use blosum or 3D vectors as well. According to our naming convention we named these descriptors as Sequential Blosum and Sequential 3D correspondingly.

3.3.2.1 Descriptor 4: Sequential Sparse (SS)

Definition 3.3.3. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Sequential Sparse descriptor of protein* P *is*

$$SS\left(P\right) = \left(S\left(p_1\right) || S\left(p_2\right) || \ldots S\left(p_n\right)\right)$$

3.3.2.2 Descriptor 5: Sequential Blosum (SB)

Definition 3.3.4. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Sequential Blosum descriptor of protein* P *is*

$$SB\left(P\right) = \left(B\left(p_1\right) || B\left(p_2\right) || \ldots B\left(p_n\right)\right)$$

3.3.2.3 Descriptor 6: Sequential 3D (ST)

Definition 3.3.5. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Sequential 3D descriptor of protein* P *is*

$$ST\left(P\right) = \left(T\left(p_1\right) || T\left(p_2\right) || \ldots T\left(p_n\right)\right)$$

3.3.2.4 Properties of Sequential Descriptors

In sequential descriptors;

- if used sparse vectors then amino acids are all orthogonal and in equal distance to each other. Therefore each amino acid is only similar to itself.

- if used blosum vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other

according to mutability rate.

- if used 3D vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to their 3D attributes.

Besides contributing amino acids' mutability and 3D attributes into the descriptors, Sequential Blosum and Sequential 3D descriptors have full sequence order effect just as the Conventional Sequential descriptor. They are not applicable to proteins with various lengths. Their output has $a.n$ dimensions where n is the protein length and a is the number dimensions of amino acid vectors.

3.3.2.5 Concretization of Sequential Descriptors on Figure

Sequential descriptors can be concreted in Figure 3.3 assuming the followings:

Each amino acid, an a dimensional vector, is represented as a ring notated in its center with number of dimensions.

A protein of n residues is represented as a sequence of n amino acid rings.

A sequential descriptor, a d dimensional vector, of a protein is represented as a sequence of n rings notated in their center with number of dimensions.

Since a sequential descriptor is concatenation of vectors of amino acids in the protein then $d = a.n$ varying with the protein length n.

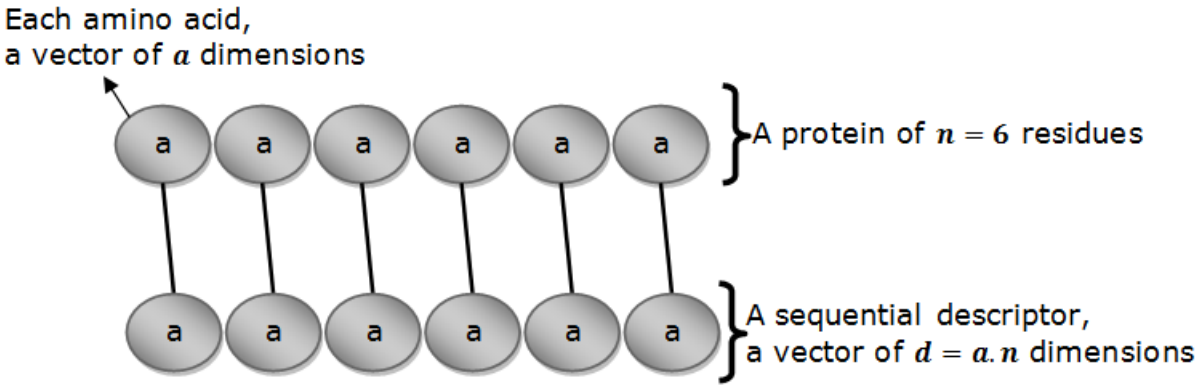

Figure 3.3: A sequential descriptor

3.3.3 Discrete Sequential Descriptors

Discrete descriptors have no full sequence order effect whereas sequential descriptors have full sequence order effect. In between them we define the hybrid descriptors with partial sequence order effect. In hybrid descriptors the parameterization is on the dimension values of amino acid vectors as well as on the partial summation and partial concatenation of these vectors. Hybrid descriptors are two; discrete sequential descriptors and sequential discrete descriptors.

Discrete sequential descriptors first split amino acid sequence, then encode each split with a discrete descriptor, then concatenate the discrete encodings of the splits. Consequently discrete sequential descriptors include partial but not full sequence order effect. A discrete descriptor to encode each split can be any of the discrete descriptors (Discrete Sparse, Discrete Blosum, Discrete 3D) defined in above. According to our naming convention we named these descriptors as Discrete Sequential Sparse, Discrete Sequential Blosum and Discrete Sequential 3D correspondingly.

3.3.3.1 Descriptor 7: Discrete Sequential Sparse (DSS)

Definition 3.3.6. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Discrete Sequential Sparse descriptor of protein* P *is*

$$DSS\,(P) = (\sum_{i=1}^{\lfloor n/s \rfloor} S\,(p_i)\,||\sum_{i=\lfloor n/s \rfloor+1}^{2\lfloor n/s \rfloor} S\,(p_i)\,||\ldots\sum_{i=(s-1)\lfloor n/s \rfloor+1}^{n} S\,(p_i))$$

where s *is the number of splits and* $s = 1, 2, 3, \ldots n$

3.3.3.2 Descriptor 8: Discrete Sequential Blosum (DSB)

Definition 3.3.7. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Discrete Sequential Blosum descriptor of protein* P *is*

$$DSB\,(P) = (\sum_{i=1}^{\lfloor n/s \rfloor} B\,(p_i)\,||\sum_{i=\lfloor n/s \rfloor+1}^{2\lfloor n/s \rfloor} B\,(p_i)\,||\ldots\sum_{i=(s-1)\lfloor n/s \rfloor+1}^{n} B\,(p_i))$$

where s *is the number of splits and* $s = 1, 2, 3, \ldots n$

3.3.3.3 Descriptor 9: Discrete Sequential 3D (DST)

Definition 3.3.8. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Discrete Sequential 3D descriptor of protein* P *is*

$$DST\,(P) = (\sum_{i=1}^{\lfloor n/s \rfloor} T\,(p_i)\,||\sum_{i=\lfloor n/s \rfloor+1}^{2\lfloor n/s \rfloor} T\,(p_i)\,||\ldots\sum_{i=(s-1)\lfloor n/s \rfloor+1}^{n} T\,(p_i))$$

where s *is the number of splits and* $s = 1, 2, 3, \ldots n$

3.3.3.4 Properties of Discrete Sequential Descriptors

In discrete sequential descriptors;

- if used sparse vectors then amino acids are all orthogonal and in equal distance to each other. Therefore each amino acids is only similar to itself.

- if used blosum vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to mutability rate.

- if used 3D vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to their 3D attributes.

Besides contributing amino acids' mutability and 3D attributes into the descriptors, discrete sequential descriptors have partial but not full sequence order effect.

If number of splits is one then they have no sequence order effect, as the number of splits increases their sequence order effect increases until the full sequence order effect when the number of splits equal to the number of amino acids that constitute the protein. They are applicable to proteins with various lengths. Their output has $a.s$ dimensions where s is the number of splits and a is the number dimensions of amino acid vectors.

3.3.3.5 Concretion of Discrete Sequential Descriptors on Figure

Discrete sequential descriptors can be concreted in Figure 3.4 assuming the followings:

Each amino acid, an a dimensional vector, is represented as a ring notated in its center with number of dimensions.

A protein of n residues is represented as a sequence of n amino acid rings.

A discrete sequential descriptor, a d dimensional vector, of a protein is represented as a sequence of r rings notated in their center with number of dimensions. Each of the r rings is sum of vectors of amino acids in each of the s splits.

Since a discrete sequential descriptor is concatenation of sum of vectors of amino acids in each of the s splits in the protein then $d = a.s$ regardless of protein length n.

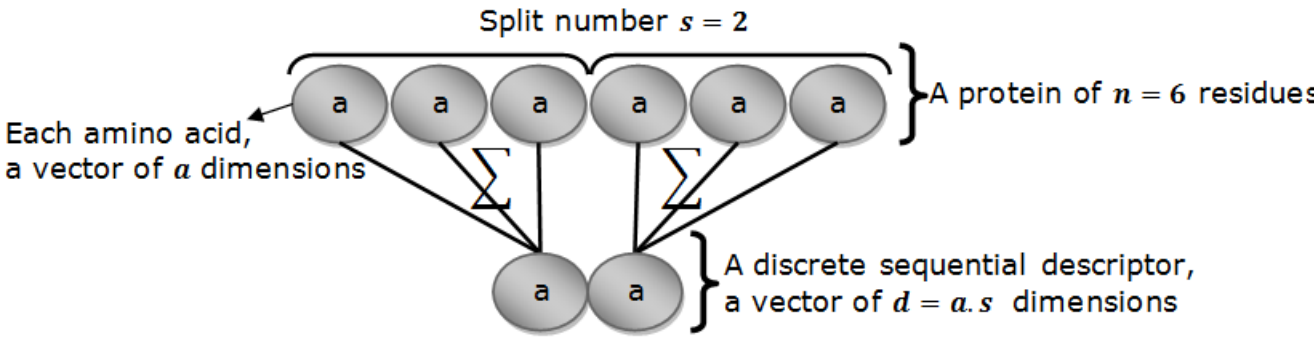

Figure 3.4: A discrete sequential descriptor

3.3.4 Sequential Discrete Descriptors

Sequential discrete descriptors first concatenates every amino acid at a specific step size, then encode the concatenations with a discrete descriptor. Consequently sequential discrete descriptors include partial but not full sequence order effect. A discrete descriptor can be any of the discrete descriptors (Discrete Sparse, Discrete Blosum, Discrete 3D) defined in above. According to our naming convention we named these descriptors as Sequential Discrete Sparse, Sequential Discrete Blosum and Sequential Discrete 3D correspondingly.

3.3.4.1 Descriptor 10: Sequential Discrete Sparse (SDS)

Definition 3.3.9. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Sequential Discrete Sparse descriptor of protein* P *is*

$$SDS\,(P) = \sum_{i=1}^{\lfloor n/s \rfloor} \left(S\left(p_{s(i-1)+1}\right) || S\left(p_{s(i-1)+2}\right) || \ldots S\left(p_{s(i-1)+s}\right)\right)$$

where s *is the number of amino acids at each step size and* $s = 1, 2, 3, \ldots n$

3.3.4.2 Descriptor 11:Sequential Discrete Blosum (SDB)

Definition 3.3.10. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Sequential Discrete Blosum descriptor of protein* P *is*

$$SDB\,(P) = \sum_{i=1}^{\lfloor n/s \rfloor} \left(B\left(p_{s(i-1)+1}\right) || B\left(p_{s(i-1)+2}\right) || \ldots B\left(p_{s(i-1)+s}\right)\right)$$

where s *is the number of amino acids at each step size and* $s = 1, 2, 3, \ldots n$

3.3.4.3 Descriptor 12: Sequential Discrete 3D (SDT)

Definition 3.3.11. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Sequential Discrete 3D descriptor of protein* P *is*

$$SDT\left(P\right) = \sum_{i=1}^{\lfloor n/s \rfloor} \left(T\left(p_{s(i-1)+1}\right) || T\left(p_{s(i-1)+2}\right) || \ldots T\left(p_{s(i-1)+s}\right)\right)$$

where s *is the number of amino acids at each step size and* $s = 1, 2, 3, \ldots n$

3.3.4.4 Properties of Sequential Discrete Descriptors

In sequential discrete descriptors;

- if used sparse vectors then amino acids are all orthogonal and in equal distance to each other. Therefore each amino acids is only similar to itself.

- if used blosum vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to mutability rate.

- if used 3D vector then amino acids are not all orthogonal and in equal distance to each other. Therefore amino acids have some similarity among each other according to their 3D attributes.

Besides contributing amino acids' mutability and 3D attributes into the descriptors, sequential discrete descriptors have partial but not full sequence order effect. If step size is one then they have full sequence order effect, as step size increases their sequence order effect decreases until no sequence order effect when step size is equal to the number of amino acids that constitute the protein. They are applicable to proteins with various lengths. Their output has $20xs$ dimensions where s is the number of amino acids in a step size.

3.3.4.5 Concretion of Sequential Discrete Descriptors on Figure

Sequential discrete descriptors can be concreted in Figure 3.5 assuming the followings:

Each amino acid, an a dimensional vector, is represented as a ring notated in its center with number of dimensions.

A protein of n residues is represented as a sequence of n amino acid rings.

A sequential discrete descriptor, a d dimensional vector, of a protein is represented as a sequence of r rings notated in their center with number of dimensions. Each of the r rings is sum of vectors of amino acids in every step of s size.

Since a sequential discrete descriptor is concatenation of sum of vectors of amino acids in every step of s size in the protein then $d = a.s$ regardless of protein length n.

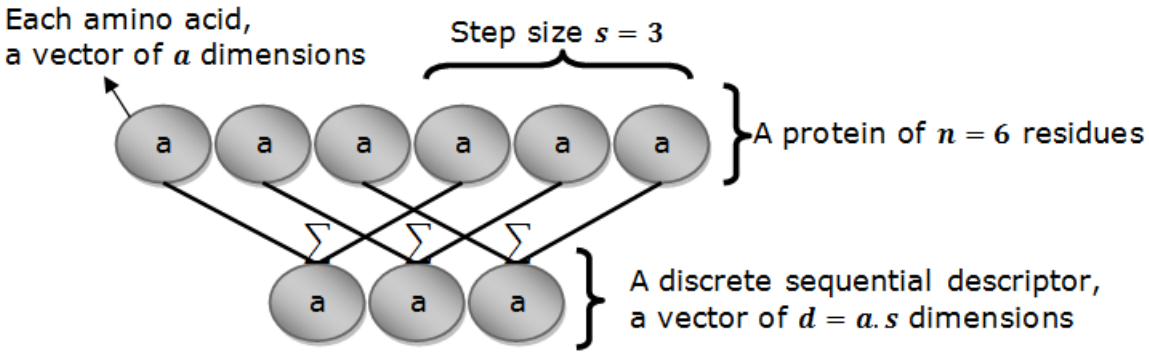

Figure 3.5: A sequential discrete descriptor

3.3.5 Miscellaneous Descriptors

Except the descriptors in our scheme (Figure 3.1) we also define the following descriptors based on our mathematical model.

3.3.5.1 Descriptor 13: Last-First Combination of Discrete Sparse (LFDS)

Last-First Combination of Discrete Sparse adds partial sequence order effect to Discrete Sparse descriptor by an additional calculation. Additional calculation is the summation of the concatenated last ten dimensions of the sparse vector of an amino acid with the first ten dimensions of the sparse vector of the next amino acid in the order. LFDS is applicable to proteins with various lengths. Its output has 40 dimensions.

Definition 3.3.12. *Given the protein* $P = (p_1, p_2, p_3, \dots p_n)$ *then Last-First Combination of Discrete Sparse descriptor of protein* P *is*

$$LFDS\left(P\right) = \sum_{i=1}^{n} S\left(p_i\right) || \sum_{i=1}^{n} \left(S\left(p_i\right)_{[11,20]} || S\left(p_{i+1}\right)_{[1,10]}\right)$$

where $\boldsymbol{V}_{[i,j]}$ *denotes subvector of elements of vector* $\boldsymbol{V}$ *from to* i^{th} *to* j^{th}.

Example 3.3.1. *Assume that we only have four types of amino acids,* A, R, N, D *where* $S(A) = (1,0,0,0), S(R) = (0,1,0,0), S(N) = (0,0,1,0)$ *and* $S(D) = (0,0,0,1)$. *Given the protein* $P = ARND$. *Then*

$LFDS(P) = (1,0,0,0)+(0,1,0,0)+(0,0,1,0)+(0,0,0,1)||(0,0,0,1)+(0,0,0,0)+(1,0,0,0,)$

$LFDS(P) = (1,1,1,1)||(1,0,0,1))$

$LFDS(P) = (1,1,1,1,1,0,0,1)$

3.3.5.2 Descriptor 14:Incremental Discrete Sparse (IDS)

Incremental Discrete Sparse descriptor adds partial sequence order effect into Discrete Sparse descriptor by increasing the non-zero element of amino acid sparse vectors at each position by one. IDS is applicable to proteins with various lengths. Its output has 20 dimensions.

Definition 3.3.13. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Incremental Discrete Sparse descriptor of protein* P *is*

$$IDS\left(P\right) = \sum_{i=1}^{n} iS\left(p_i\right)$$

Example 3.3.2. *Assume that we only have four types of amino acids,* A, R, N, D *where* $S(A) = (1,0,0,0), S(R) = (0,1,0,0), S(N) = (0,0,1,0)$ *and* $S(D) = (0,0,0,1)$. *Given the protein* $P = ARND$. *Then*

$IDS(P) == (1,0,0,0) + (0,2,0,0) + (0,0,3,0) + (0,0,0,4)$

$IDS(P) = (1,2,3,4)$

3.3.5.3 Descriptor 15: Shifted Discrete Sparse (SDS)

Shifted Discrete Sparse descriptor adds partial sequence order effect into Discrete Sparse descriptor by shifting the amino acid sparse vector by one dimension to the right and filling the gaps with zeros. SDS is not applicable to proteins with various lengths. Its output has $20 + n - 1$ dimensions where n is the protein length.

Definition 3.3.14. *Given the protein* $P = (p_1, p_2, p_3, \ldots p_n)$ *then Shifted Discrete Sparse descriptor of protein* P *is*

$$SDS\left(P\right) = \sum_{i=1}^{n} \left(v_1, v_2, \ldots v_{i-1}, S\left(p_i\right), v_1, v_2, \ldots v_{n-i}\right)$$

$$v_i = 0, \ \forall i$$

Example 3.3.3. *Assume that we only have four types of amino acids,* A, R, N, D *where* $S(A) = (1,0,0,0), S(R) = (0,1,0,0), S(N) = (0,0,1,0)$ *and* $S(D) = (0,0,0,1)$. *Given the protein* $P = AAA$. *Then*

$SDS(P) = (1,0,0,0,0,0) + (0,1,0,0,0,0) + (0,0,1,0,0,0)$

$SDS(P) = (1,1,1,0,0,0)$

Chapter 4

Datasets

Protein datasets are characterized with various biological properties such as compositional, sequential, structural, evolutional, physicochemical or a hybrid of their subset. Protein descriptors describe proteins with various features such as compositional, sequential, structural, evolutional, physicochemical or a hybrid of their subset. Therefrom we had the hypothesis that a single descriptor might not be optimal for all protein datasets. In order to support our hypothesis we used datasets that diversify in their biological natures. The three datasets we used are caspase dataset (CSP), DNA binding dataset (DNA) and subcellular localization dataset (SC). As it would be recognized from our experimental results CSP dataset is characterized by sequential features of its peptides. On the other hand SC dataset is characterized by compositional features of its proteins [23]. And in between these two characters is the DNA dataset which is characterized by compositional features of a part of its proteins [1].

We can formulate the problems on these datasets in a general form as in the following.

A protein sequence dataset that contains C number of categories for the attribute to be classified is the union of subsets for each category.

$$S = S_1 \cup S_2 \cup \ldots S_C$$

Where S is the protein sequence dataset, S_i is subset of each category and $i = 1, 2, 3, \ldots C$. All our problems are binary classification where $C = 2$.

In all our experiments datasets are separated into training dataset S_L and testing dataset S_T. S_L and S_T are independent where

$$S = S_L \cup S_T$$

$$\varnothing = S_L \cap S_T$$

and stratified so they contain approximately the same proportions of labels as original dataset S. This is necessary for better scheme in variance and bias [16].

The training set is used to train the classification model and determine its optimal parameters by 3-fold cross validation and the testing set is used for evaluating the model's performance.

We used the following three datasets to evaluate the proposed descriptors. The details of each dataset are reported in Table 4.1.

Table 4.1: The details of the datasets used in our experiments

Abbreviation	**# Classes**	**Max # residues**	**Min # residues**	**# samples**
SC	2	5000	50	620
DNA	2	639	26	371
CSP	2	14	14	494

4.1 Caspase Dataset (CSP)

Caspases are cysteine proteases. A protease is an enzyme that performs proteolysis, which is the breakdown of proteins into smaller polypeptides. This dataset includes 494 protein sequences classified into positive and negative peptides. Among the 494 proteins, 247 belong to positive class; 247 belong to negative class. All peptides have fixed lengths of 14 residues. [2]

4.2 DNA Binding Dataset (DNA)

DNA binding proteins have important roles in activities associated with DNA such as DNA packaging, replication, and transcription regulation. DNA binding proteins have different functions hence different amino acid sequences and three dimensional structures which make their prediction challenging [19]. (Ahmad et al. 2004) analysed the relationship between DNA binding and protein sequence composition. They stated that the composition of binding regions differs significantly from non-binding regions, but this variance does not extend to the total composition of the protein sequence.

This dataset includes 371 protein sequences classified into DNA binding proteins or non DNA binding proteins. Among the 371 proteins, 145 belong to DNA binding class and 226 belong to non DNA binding class. Proteins have lengths vary from 26 residues to 639 residues. None of the proteins have more than 25% sequence identity to any other protein in the same class subset [19].

4.3 Subcellular Localization Dataset (SC)

Subcellular localization is important in biology because the function of protein correlates with its subcellular location. An early research performed by (Chou et al. 1999) showed that there is a contact between the protein surface and the cellular environment. Different locations of a cell have different physicochemical environments. Consequently each location accommodates a protein according to its surface physicochemical character, which is correlated with its amino acid composition [23].

This dataset includes 620 protein sequences classified into two human subcellular locations. Among the 620 proteins, 314 belong to mitochondrion location; 306 belong to extracell location. Protein length varies from 50 residues to 5000 residues. None of the proteins have more than 25% sequence identity to any other protein in the same subcellular location subset [9].

Chapter 5

Experiments and Results

This chapter discusses various experiments carried out to evaluate our descriptors versus some traditional descriptors. First, datasets are divided into training set, validation set, and test set. We further present the test results obtained by two classifiers using our descriptors and some traditional descriptors. Finally we discuss on the implication of results in biology domain.

5.1 Experiments

5.1.1 Evaluation

Each of the three datasets is used to build two classifiers, SVM and RF, using the proposed descriptors described in Section 3.3 and the traditional descriptors AAC, PAAC, APAAC, SS, SB and ST. AUC is used to measure prediction performance. Furthermore, the difference of a classifier's training ACC and testing ACC is used as an overfitting indicator. We selected SVM and RF as they are widely considered the state of the art classifiers [13]. Moreover they process data points as vectors, which fits with our mathematical modeling assumption that amino acids are high dimensional vectors. The goal of using two classifiers is to ensure that results do not reflect the classifier ability but the descriptor ability. We did not aim at comparing

the classifiers but to compare descriptors' predictive performances independent of classifiers. That is why we will not focus on the performance differences caused by classifiers.

Commonly used evaluation measure ACC is biased due to its vulnerability to class skewing [27]. Conversely AUC decouples classifier performance from class skewness [12]. For this reason we used AUC as a performance indicator. But ACC can be an indicator of overfitting. If very high training ACC is a result of overfitting, then this will lead to a low testing ACC [26]. Hence we used the difference of training ACC and testing ACC as an overfitting indicator. 3-fold cross validation is applied to evaluate the predictive performance of descriptors in terms of AUC. K-fold cross validation guarantee that each training example is trained k-1 times and tested once. Therefore it reduces the variance while increasing the bias [28]. However stratified cross validation gives better scheme in variance and bias [16], hence used in our work.

5.1.2 Building and Tuning Classifiers

The parameters of each classifier are tuned to obtain optimal classifier model. We use the automatic searching method in R package Caret [17] that uses grid search to tune the parameters and cross validation to compare their affect on the performance measure AUC.

SVM with RBF kernel has two parameters. First is σ the width of the kernel function, and the other is c the cost value. RF has a single parameter that is k the number of trees in the forest. We used Caret package to make a grid search to find the best parameters for each classifier. It first creates a grid of possible parameter combinations. Then using cross validation it creates a set of training sets and corresponding validation sets. For each candidate parameter combination, it fits a classifier to each training set and is used to predict corresponding validation set and an average of the performance measure AUCs is calculated for each of the

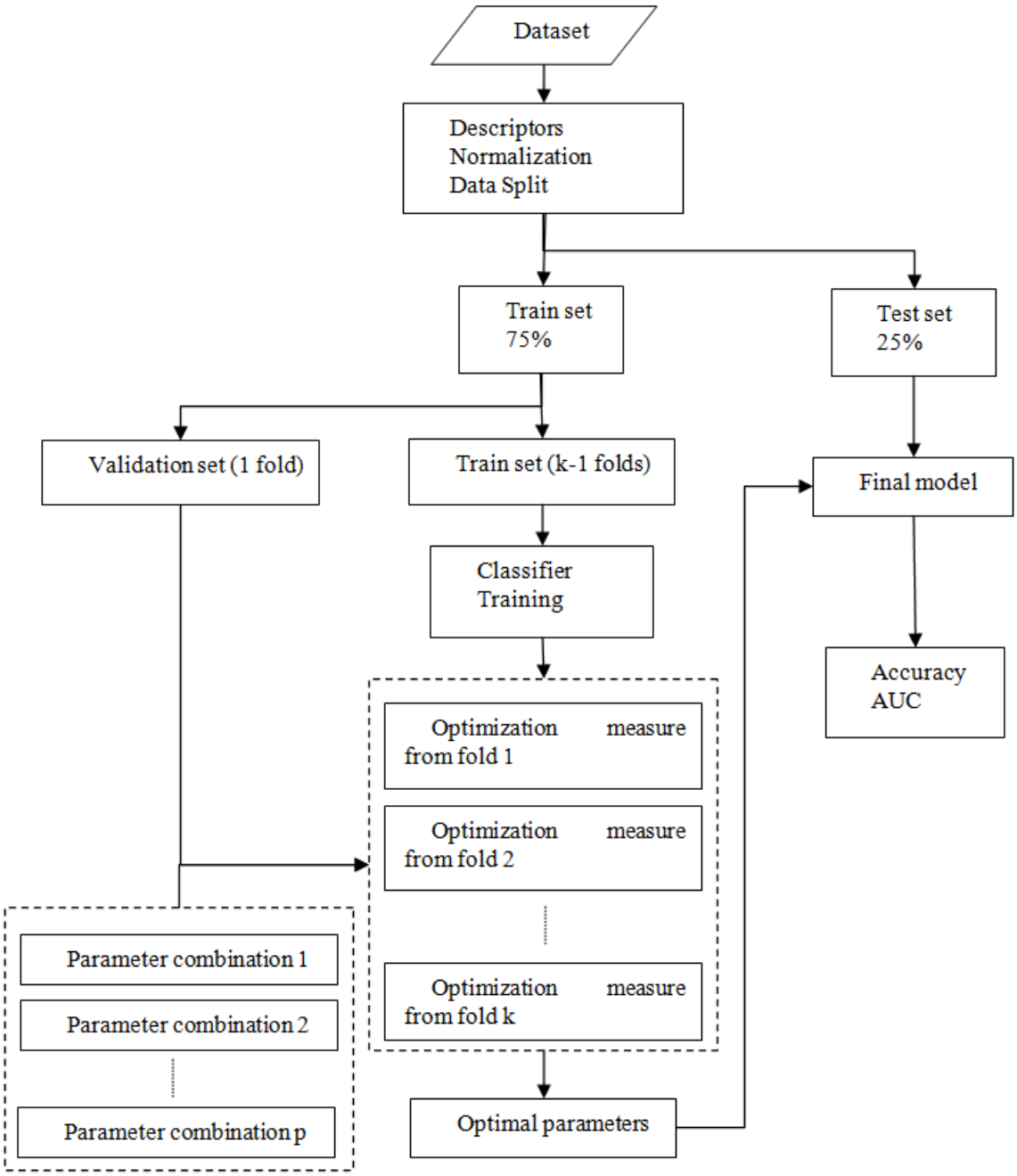

Figure 5.1: Flow chart of building and tuning classifiers

parameter combination. Finally it uses averaged AUC values to select the optimal parameter combination. Once optimal parameters are assigned, final model with the optimal parameters is used to predict test set and to calculate performance measures AUC and ACC (Figure 5.1). Caret package implements this procedure

with the following algorithm [18].

```
Define sets of model parameter values to evaluate
for each parameter set do
   for each cross validation fold do
      hold out specific samples
      fit the model on the remainder
      predict the hold-out samples
   end
   calculate the average performance
end
determine the optimal parameter set
fit the model to all the training data using optimal parameters
```

5.2 Results

We performed three sets of experiments. Each set of experiments is performed on one dataset. We used the following naming conventions in the result tables. Descriptors proposed and modeled by this thesis are indicated in bold text.

1. AAC: Amino Acid Composition
2. PAAC: Pseudo Amino Acid Composition
3. APAAC: Amphiphilic Pseudo Amino Acid Composition by [7]. It is a version of PAAC that considers amino acids' amphiphilic properties.
4. **DT**: Discrete 3D
5. **DB**: Discrete Blosum
6. **IDS**: Incremental Discrete Sparse
7. **LFDS**: Last First Combination of Discrete Sparse

8. **DSS2Split**: Discrete Sparse with 2 splits
9. **DSS3Split**: Discrete Sequential Sparse with 3 splits
10. **DSS4Split**: Discrete Sequential Sparse with 4 splits
11. **DSS5Split**: Discrete Sequential Sparse with 5 splits
12. **DSB2Split**: Discrete Sequential Blosum with 2 splits
13. **DST2Split**: Discrete Sequential 3D with 2 splits
14. **SDS2Step**: Sequential Discrete Sparse with 2 steps
15. **SDB2Step**: Sequential Discrete Blosum with 2 steps
16. **SDT2Step**: Sequential Discrete 3D with 2 steps
17. SS: Sequential Sparse
18. SB: Sequential Blosum
19. ST: Sequential 3D
20. **SDS**: Shifted Discrete Sparse

The results of discrete sequential descriptors were impressive. We continued increasing the split number of Discrete Sequential Sparse descriptor until no more improvement in predictive performance.

Notice that the descriptors SS, ST and SDS are not included in the experiments on SC and DNA datasets but on CSP dataset. This is because they are applicable to fixed length proteins namely peptides. The other descriptors however are useful for various length of proteins.

5.2.1 Results on SC Dataset

First set of experiments are aimed at comparing proposed descriptors in chapter 3 with the traditional descriptors AAC, PAAC, APAAC, SS, SB and ST on SC

dataset. For a fair comparison; the same classifier, same parameter optimization algorithm and same evaluation measures is used with each of the descriptors.

In Table 5.1 number of dimensions, AUC, training ACC, testing ACC, difference of training and testing ACCs, encoding time (in seconds) and modelling time (in seconds) of the descriptors obtained by RF classifier on SC dataset is reported.

Table 5.1: Results on SC dataset by RF classifier

	dimensions	auc	accTr	accTe	accTr-accTe	enctime	modeltime
AAC	20	0.903	0.798	0.818	-0.019	1.45	8.939
PAAC	50	0.856	0.77	0.772	-0.002	185.779	17.68
APAAC	80	0.809	0.759	0.74	0.019	149.159	25.549
DT	3	0.66	0.566	0.577	-0.011	1.36	1.76
DB	20	0.793	0.755	0.714	0.041	1.469	9.61
IDS	40	0.878	0.785	0.831	-0.045	4.159	14.14
LFDS	40	0.932	0.763	0.883	-0.12	26.909	14.76
DSS2Split	40	0.927	0.8	0.831	-0.03	1.639	14.299
DSS3Split	60	0.887	0.845	0.831	0.014	1.82	18.619
DSS4Split	80	0.922	0.838	0.844	-0.006	1.99	23.319
DSS5Split	100	0.974	0.83	0.915	-0.085	2.239	28.459
DSB2Split	40	0.87	0.787	0.772	0.015	1.7	14.629
DST2Split	6	0.757	0.65	0.707	-0.056	1.67	3.77
SDS2Step	40	0.905	0.778	0.811	-0.033	3.81	14.959
SDB2Step	40	0.863	0.712	0.772	-0.06	3.82	15.889
SDT2Step	6	0.621	0.631	0.584	0.047	3.62	4.069

In Table 5.2 number of dimensions, AUC, training ACC, testing ACC, difference of training and testing ACCs, encoding time (in seconds) and modelling time (in seconds) of the descriptors obtained by SVM classifier on SC dataset is reported.

5.2.2 Results on DNA Dataset

Second set of experiments are aimed at comparing the proposed descriptors in chapter 3 with the traditional descriptors AAC, PAAC, APAAC, SS, SB and ST on DNA dataset. For a fair comparison the same classifier, same parameter optimization algorithm, and same evaluation measures are used with each of the descriptors.

Table 5.2: Results on SC dataset by SVM classifier

	dimensions	auc	accTr	accTe	accTr-accTe	enctime	modeltime
AAC	20	0.902	0.774	0.824	-0.049	1.459	4.89
PAAC	50	0.807	0.736	0.733	0.003	186.059	4.709
APAAC	80	0.816	0.723	0.74	-0.017	149.44	5.559
DT	3	0.658	0.618	0.636	-0.018	1.51	3.329
DB	20	0.85	0.811	0.779	0.032	1.48	3.6
IDS	40	0.924	0.802	0.857	-0.054	4.11	4.219
LFDS	40	0.877	0.791	0.766	0.025	27.59	4.349
DSS2Split	40	0.917	0.821	0.863	-0.042	1.63	4.31
DSS3Split	60	0.905	0.847	0.811	0.035	1.88	4.959
DSS4Split	80	0.916	0.858	0.85	0.008	2.059	5.459
DSS5Split	100	0.918	0.843	0.844	-0.001	2.239	6.119
DSB2Split	40	0.909	0.792	0.811	-0.019	1.699	4.239
DST2Split	6	0.702	0.635	0.642	-0.007	1.569	3.42
SDS2Step	40	0.839	0.776	0.753	0.023	3.69	4.409
SDB2Step	40	0.868	0.761	0.792	-0.031	3.77	4.639
SDT2Step	6	0.626	0.592	0.564	0.028	3.719	3.549

In Table 5.3 number of dimensions, AUC, training ACC, testing ACC, difference of training and testing ACCs, encoding time (in seconds) and modelling time (in seconds) of the descriptors obtained by RF classifier on DNA dataset is reported.

Table 5.3: Results on DNA dataset by RF classifier

	dimensions	auc	accTr	accTe	accTr-accTe	enctime	modeltime
AAC	20	0.856	0.792	0.75	0.042	1	9.18
PAAC	45	0.822	0.749	0.76	-0.011	53.099	8.239
APAAC	70	0.863	0.72	0.793	-0.073	43.62	11.86
DT	3	0.749	0.688	0.706	-0.018	0.809	1.09
DB	20	0.833	0.767	0.739	0.028	0.869	4.74
IDS	40	0.878	0.781	0.771	0.01	1.34	7.689
LFDS	40	0.894	0.745	0.804	-0.059	19.329	7.319
DSS2Split	40	0.964	0.756	0.913	-0.157	0.95	7.669
DSS3Split	60	0.897	0.781	0.815	-0.033	1.08	10.28
DSS4Split	80	0.837	0.777	0.75	0.027	1.189	12.759
DSS5Split	100	0.843	0.781	0.706	0.075	1.31	14.9
DSB2Split	40	0.896	0.734	0.826	-0.091	1	7.76
DST2Split	6	0.705	0.72	0.63	0.089	0.919	2.08
SDS2Step	40	0.847	0.759	0.76	-0.001	1.44	7.729
SDB2Step	40	0.84	0.77	0.75	0.02	1.47	7.84
SDT2Step	6	0.829	0.666	0.76	-0.093	1.389	2.34

In Table 5.4 number of dimensions, AUC, training ACC, testing ACC, difference of training and testing ACCs, encoding time (in seconds) and modelling time (in seconds) of the descriptors obtained by SVM classifier on DNA dataset is reported.

Table 5.4: Results on DNA dataset by SVM classifier

	dimensions	auc	accTr	accTe	accTr-accTe	enctime	modeltime
AAC	20	0.844	0.806	0.771	0.035	0.87	2.479
PAAC	45	0.846	0.734	0.771	-0.037	52.7	2.889
APAAC	70	0.801	0.702	0.717	-0.015	41.79	14.229
DT	3	0.792	0.641	0.673	-0.032	0.81	2.309
DB	20	0.858	0.748	0.804	-0.056	0.889	2.43
IDS	40	0.803	0.802	0.717	0.085	1.389	2.819
LFDS	40	0.891	0.774	0.793	-0.019	15.88	2.809
DSS2Split	40	0.894	0.784	0.804	-0.02	0.979	2.84
DSS3Split	60	0.881	0.774	0.782	-0.008	1.09	3.179
DSS4Split	80	0.812	0.799	0.75	0.049	1.22	3.5
DSS5Split	100	0.923	0.777	0.858	-0.08	1.34	3.819
DSBT2Split	40	0.84	0.727	0.793	-0.066	1	2.77
DST2Split	6	0.683	0.71	0.663	0.046	0.94	2.389
SDS2Step	40	0.868	0.785	0.793	-0.008	1.45	4.359
SDB2Step	40	0.825	0.734	0.717	0.017	1.5	2.9
SDT2Step	6	0.778	0.716	0.673	0.042	1.439	4.02

5.2.3 Results on CSP Dataset

Third set of experiments are aimed at comparing proposed descriptors in chapter 3 with the traditional descriptors AAC, PAAC, APAAC, SS, SB and ST on DNA dataset. For a fair comparison the same classifier, same parameter optimization algorithm and same evaluation measures is used with each of the descriptors.

In Table 5.5 number of dimensions, AUC, training ACC, testing ACC, difference of training and testing ACCs, encoding time (in seconds) and modelling time (in seconds) of the descriptors obtained by RF classifier on CSP dataset is reported.

In Table 5.6 number of dimensions, AUC, training ACC, testing ACC, difference of training and testing ACCs, encoding time (in seconds) and modelling time (in seconds) of the descriptors obtained by SVM classifier on CSP dataset is reported.

Table 5.5: Results on CSP dataset by RF classifier

	dimensions	auc	accTr	accTe	accTr-accTe	enctime	modeltime
AAC	20	0.707	0.666	0.639	0.027	1.15	7.459
PAAC	30	0.715	0.693	0.647	0.045	9.689	9.17
APAAC	40	0.756	0.717	0.704	0.013	9.5	11.169
DT	3	0.662	0.577	0.606	-0.029	1.069	1.36
DB	20	0.751	0.607	0.68	-0.073	1.139	7.139
IDS	40	0.813	0.752	0.704	0.048	0.25	10.47
LFDS	40	0.746	0.637	0.696	-0.058	21.56	11.18
DSS2Split	40	0.822	0.766	0.737	0.029	1.27	10.389
DSS3Split	60	0.821	0.72	0.721	-0.001	1.439	14.11
DSS4Split	80	0.812	0.803	0.754	0.049	1.59	17.329
DSS5Split	100	0.876	0.822	0.811	0.01	1.729	19.919
DSB2Split	40	0.795	0.741	0.696	0.045	1.359	11.12
DST2Split	6	0.668	0.62	0.614	0.006	1.209	3
SDS2Step	40	0.751	0.739	0.696	0.043	1.389	10.619
SDB2Step	40	0.839	0.693	0.778	-0.085	1.389	11.18
SDT2Step	6	0.7	0.623	0.614	0.009	1.29	3
SS	280	0.915	0.844	0.836	0.008	1.139	43.58
SB	280	0.932	0.854	0.836	0.018	1.189	45.979
ST	42	0.959	0.849	0.893	-0.044	0.979	9.739
SDS	34	0.777	0.669	0.704	-0.034	1.149	9.86

5.2.4 Discussion

The aim of the experiments was to compare the behaviour of traditional and novel protein descriptors, in order to investigate the effectiveness of novel descriptors and point out the importance of choosing the right descriptor for a biological problem. In each dataset the best performing descriptor was different as we expected in our hypothesis. In SC and DNA datasets, DSS descriptor was the best performing descriptor. In CSP dataset, SS and ST were the best performing descriptors.

Looking at the Figure5.2 sequential descriptor ST is the best performing descriptor on CSP dataset however it is not applicable to DNA and SC datasets due to their variable length proteins. Partial discrete descriptor DSS2Split is the best performing descriptor on DNA dataset however it is not as good as ST on CSP dataset and as DSS5Split on SC dataset. SC dataset is best described by another

Table 5.6: Results on CSP dataset by SVM classifier

	dimensions	auc	accTr	accTe	accTr-accTe	enctime	modeltime
AAC	20	0.733	0.669	0.68	-0.011	1.19	3.319
PAAC	30	0.765	0.677	0.713	-0.035	9.83	3.48
APAAC	40	0.751	0.741	0.68	0.06	9.86	3.679
DT	3	0.662	0.661	0.581	0.08	1.099	2.809
DB	20	0.73	0.65	0.68	-0.03	1.209	3.2
IDS	40	0.782	0.65	0.737	-0.086	0.279	3.68
LFDS	40	0.723	0.682	0.631	0.051	21.94	3.93
DSS2Split	40	0.771	0.784	0.704	0.08	1.34	3.709
DSS3Split	60	0.82	0.717	0.704	0.013	1.48	4.119
DSS4Split	80	0.868	0.723	0.795	-0.072	1.81	4.43
DSS5Split	100	0.911	0.782	0.852	-0.069	1.809	4.809
DSB2Split	40	0.796	0.712	0.696	0.016	1.379	3.69
DST2Split	6	0.693	0.645	0.672	-0.027	1.29	2.949
SDS2Step	40	0.805	0.731	0.704	0.027	1.43	3.919
SDB2Step	40	0.785	0.666	0.721	-0.054	1.5	3.84
SDT2Step	6	0.748	0.653	0.704	-0.05	1.339	2.919
SS	280	0.944	0.811	0.877	-0.065	1.19	8.75
SB	280	0.915	0.852	0.836	0.016	1.25	8.9
ST	42	0.858	0.803	0.803	0	1.019	3.599
SDS	34	0.832	0.677	0.778	-0.1	1.209	3.51

partial discrete descriptor, DSS5Split.

Once we have the right and related input data, features encoding and selection is paramount and critical factor. Features have to express data without losing relevant information. That is why it is necessary to search for descriptors that are suitable. the machine learning algorithm is not the only main factor of predictive performance. In this thesis we proposed new descriptors with fixed lengths that encode various characters of proteins. Hence we had more descriptors to encode various length proteins. Then we used all, the traditional and the proposed, descriptors on three datasets by comparing their performance. The aim is to reason if selecting a proper descriptor for protein classification problem is important or not.

The performance of the descriptors are ranked by AUC values of the classifier. Accordingly the performance of the descriptors are categorized into three groups

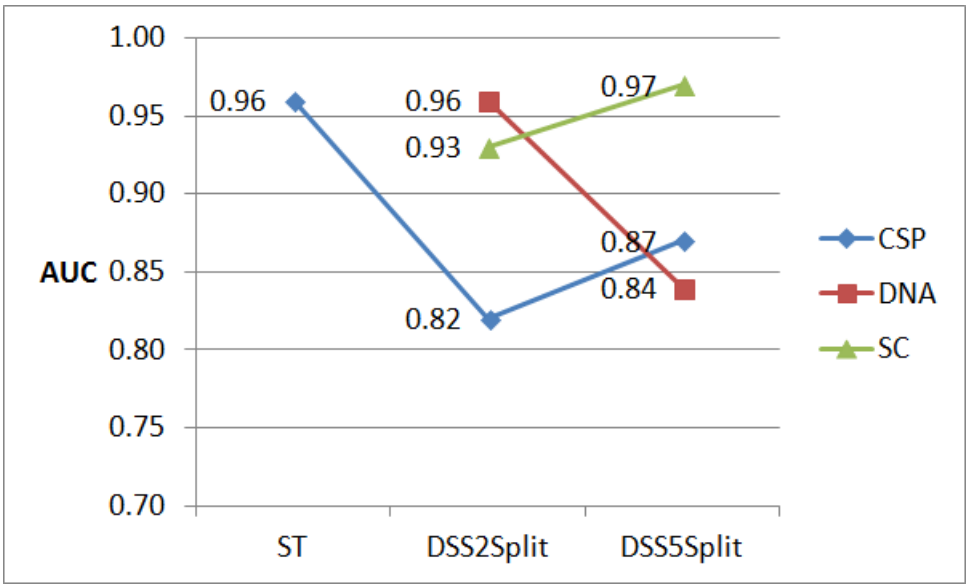

Figure 5.2: Line chart of the best descriptors ST, DSS2Split and DSS2split on CSP, DNA and SC dataset correspondingly.

based on their AUC values: 'very good' (greater than 0,85), 'good' (less than 0,85 and greater than 0,80), 'bad' (less than 0,80). The overfitting of the descriptors are ranked by the difference of training ACC and testing ACC. Accordingly the overfitting of the descriptors are categorized into two groups based on their difference values of training ACC and testing ACC: 'acceptable' (less than 0,04) and 'unacceptable' (greater than 0,04).

5.2.4.1 Discussion on SC Dataset Results

The first set of experiments on SC dataset are performed using RF. Table 5.1 shows that all descriptors except DB, DT, DST2Split and SDT2Step capture the subcellular location space of proteins in either good or very good levels. It also shows that all descriptors' overfitting indicators, difference of training ACC and testing ACC, are in acceptable levels. The best performing descriptors are DSS5Split (AUC 0.974), LFDS (AUC 0,932) and AAC (AUC 0.903). Given that subcellular locations are determined by amino acid composition of whole protein sequence, a good performance was expected from AAC. However as we continue increasing the

split numbers in DSS descriptor, performance continue to increase until 5 splits. As we discussed in methodology, increasing split number of DS descriptors increases the sequence order information they contain. The LFDS is performing better than AAC as well. This is because LFDS and DSS contain what AAC contains, the compositional information, and even more the partial sequence order effect. Then we can conclude that SC dataset proteins are characterized by AAC as well as by partial sequence order information among their residues.

The IDS (AUC 0,878) and SDS2Step (AUC 0,905) perform better than PAAC (AUC 0,856) and its variation APAAC (AUC 0.809) although their encoding times (4,159 s and 3,81 s respectively) are significantly better than PAAC's and APAAC's (185,779 s and 149,159s respectively). The IDS and SDS do not contain what AAC contain, the compositional information, but built on completely partial sequence order effect. However PAAC and APAAC contain AAC in their first 20 dimensions and they could not perform as well as IDS and SDS. Then we can conclude that complex calculations used in PAAC and APAAC to include sequence order effect does not increase the performance but the encoding time.

Second set of experiments on SC dataset are performed using SVM. The results have similar trend as discussed above. Table 5.2 shows that all descriptors except DT, SDS2Step and SDT2Step capture the subcellular location space of proteins either in good or very good levels. It also shows that all descriptors' overfitting indicators, difference of training ACC and testing ACC, are in acceptable levels. The best performing descriptors are IDS (AUC 0,924), DSS5Split (AUC 0.918), and AAC (AUC 0.902). Given that subcellular locations are determined by amino acid composition of the whole protein sequence, a good performance was expected from AAC. However the results from SVM show that including partial sequence order effect with AAC does not increase the performance significantly. We believe this is because the increased number of dimensions. Since RF has its own feature selection mechanism, increased number of dimensions does not prevent performance increment.

The IDS (AUC 0,924) and SDB2Step (AUC 0,868) perform significantly better than PAAC (AUC 0,807) and its variation APAAC (AUC 0.816) although their encoding times (4,11 s and 3,77 s respectively) are significantly better than PAAC's and APAAC's (185,059 s and 149,44 s respectively). The IDS and SDS do not contain what AAC contain but built on completely partial sequence order effect. However PAAC and APAAC contain AAC in their first 20 dimensions and they could not perform as well as IDS and SDS. Then we can conclude that complex calculations used in PAAC and APAAC to include sequence order effect do not increase the performance but the encoding time.

5.2.4.2 Discussion on DNA Dataset Results

The first set of experiments on DNA dataset are performed using RF. Table 5.3 shows that all descriptors except DT and DST2Step capture the DNA binding sites of proteins in either good or very good levels. It also shows that all descriptors' overfitting indicators, difference of training ACC and testing ACC, are in acceptable levels, except the DSS5Split (0,075) and the DST2Split (0,089). The rationale here was that as DSS split number increases the descriptor overfits and AUC decreases and DST2Split had already low AUC value. RF has well generalization ability due to its ensemble nature. Hence the pattern obtained from these descriptors do not clearly separate the the binding and non-binding proteins in the space, explaining their overfitting. The best performing descriptors are DSS2Split (AUC 0.964), LFDS (AUC 0,894), IDS (AUC 0,878) and AAC (AUC 0.856). Given that DNA binding sites are determined by amino acid composition of only the binding site but not the whole protein, a good performance was not expected from AAC. DSS2Split obtained the best performance. The LFDS and the IDS are performing better than AAC as well. This is because DNA dataset proteins' only binding sites are characterized by amino acid composition. It is likely that some patterns from sequence order effect contributes the performance.

The APAAC (AUC 0.863) performs slightly better than AAC (AUC 0,856).In-

cluding sequence order information contributed as expected but . But its performance is not as good as IDS (AUC 0.878) and LFDS (AUC 0.894) although its encoding (43,62 s) is very slow in comparison to IDS and LFDS (1,34 s and 19,329 s respectively). Hence complex calculations used in PAAC and APAAC to include sequence order effect do not increase the performance in expected levels but the encoding time.

Second set of experiments on DNA dataset are performed using SVM. Results do not have similar trend as discussed above. Table 5.4 shows that all descriptors except DT, DST2Split and SDT2Step capture the DNA binding sites of proteins in either good or very good levels. It also shows that all descriptors' overfitting indicators, difference of training ACC and testing ACC, are in acceptable levels except the IDS (0.085). However the IDS space of binding and non-binding proteins was well separated with RF. Probably SVM is not able separate the space of binding and non-binding space described by the IDS. To gain a further understanding of this overfitting, an analysis with various types of kernels is needed. The best performing descriptors are DSS5Splits (AUC 0,923), DSS2Split (AUC 0.894), and LFDS (AUC 0.891). In opposed to results with RF, increasing the DSS split number first decreases, then increases the performance. As if there are local minimums therefore it would be advisable to use a search method for best number of splits.

The LFDS (AUC 0,891) performs significantly better than PAAC (AUC 0,846) and its variation APAAC (AUC 0.801) although its encoding time (15,88 s) is significantly better than PAAC's and APAAC's (52,7 s and 41,79 s respectively). Than we can conclude that complex calculations used in PAAC and APAAC to include sequence order effect do not increase the performance as much as LFDS but the encoding time.

5.2.4.3 Discussion on CSP Dataset Results

While the above SC and DNA datasets are composed of various length of proteins, the CSP dataset employed in this thesis is composed of fixed length peptides and

hence we included the dataset. Moreover this set has been very difficult to model with discrete descriptors and is hence a very good example of the cases where sequential descriptors are successful.

At first glance on the two sets of experiments with RF 5.6 and SVM 5.6, we can observe that CSP dataset is best discriminated by sequential descriptors, ST with RF (AUC 0,959) and SS with SVM (AUC 0,944). This hypothesis can be supported by observing the increased performance with increased number of DSS splits. Furthermore low performance values of the AAC with RF (AUC 0,707) and the AAC with SVM (AUC 0,733) validate the same hypothesis.

Notice that the SDS descriptor is employed only in CSP dataset because it is not applicable to various lengths of proteins. However its performance (0,832) is not as well as the sequential descriptors. Shown that CSP dataset is merely discriminated by full sequence order effect, the SDS with partial sequence order effect partially compensates the representation of CSP proteins.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis we set a novel mathematical definition for the conventional discrete descriptor. Based on this definition we attempted to define novel fixed length protein descriptors that consider amino acid mutability, or 3-D amino acid attributes. At the same time these descriptors represent compositional information or sequential information of a protein sequence or both. Having more descriptors to use with various length of proteins, we observed the effect using various classifiers and various datasets on the predictive performance of the proposed descriptors.

From the results obtained we can conclude that the descriptors proposed in this thesis, which comprises the mathematical definition of conventional discrete descriptor, usually return good results and do not differ significantly from traditional descriptors. In particular descriptors based on sparse and blosum matrices tend to give better prediction performance than Pseudo Amino Acid Composition. In addition all the novel descriptors are significantly more efficient than Pseudo amino Acid Composition in terms of execution time. We can also state that there is no one descriptor that could be utilized for all datasets because prediction performance is dependent on the dataset's biological properties. Some datasets contain

proteins that are characterized by their amino acid composition; some datasets contain proteins that are characterized by their amino acid order and others in between both. Performance of protein classification can be enhanced by considering their characterization and selecting descriptor that can be the best fit for the classification algorithm.

We believe that the proposed descriptors will be useful in exploring meaningful encodings in protein datasets. This provides useful information for biologists to use computational methods for vast number of biological data.

6.2 Future Work

In this section we discuss possible further work that can be studied in future. In this work we studied sparse, blosum and 3D amino acid vectors. Sparse vectors represents no similarity among the amino acids. Blosum vectors represents some similarity among the amino acids based on their mutability rate to each other. 3D vectors are dimensionally reduced blosum vectors that best explain them in the three dimensional space. Each amino acid representation lead to significant predictive performance in a specific dataset. However any other type of amino acid vectors can be used instead of the above three in the proposed descriptors. For example blosum vectors are derived by analyzing the mutation frequencies seen in the block sequence alignments of known protein families [14]. On the other hand mutation frequencies in the block sequence alignment of the dataset under experiment are different than of the known protein families.

Future work may tackle this problem to find the best amino acid vectors that represent the proteins in a specific dataset. This is a search problem and can be solved using genetic algorithm with amino acid vector dimensions are the chromosomes and AUC obtained in classification problem is the objective function.

Any type of amino acid descriptor can be used in our novel protein descriptors. Another future work may experiment the predictive performance of the amino acid

descriptors other then sparse, blosum and 3D.

Bibliography

[1] Shandar Ahmad, M Michael Gromiha, and Akinori Sarai. Analysis and prediction of dna-binding proteins and their binding residues based on composition, sequence and structural information. Bioinformatics, 20(4):477–486, 2004.

[2] Muneef Ayyash, Hashem Tamimi, and Yaqoub Ashhab. Developing a powerful in silico tool for the discovery of novel caspase-3 substrates: a preliminary screening of the human proteome. BMC bioinformatics, 13(1):14, 2012.

[3] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition, 30(7):1145–1159, 1997.

[4] Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[5] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. wadsworth. Belmont, CA, 1984.

[6] Kuo-Chen Chou. Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins: Structure, Function, and Bioinformatics, 43(3):246–255, 2001.

[7] Kuo-Chen Chou. Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. Bioinformatics, 21(1):10–19, 2005.

[8] Kuo-Chen Chou. Some remarks on protein attribute prediction and pseudo amino acid composition. Journal of theoretical biology, 273(1):236–247, 2011.

[9] Kuo-Chen Chou and Hong-Bin Shen. Cell-ploc: a package of web servers for predicting subcellular localization of proteins in various organisms. Nature protocols, 3(2):153–162, 2008.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine learning, 20(3):273–297, 1995.

[11] Sean R Eddy. Where did the blosum62 alignment score matrix come from? Nature biotechnology, 22(8):1035–1036, 2004.

[12] Tom Fawcett. An introduction to roc analysis. Pattern recognition letters, 27(8):861–874, 2006.

[13] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? The Journal of Machine Learning Research, 15(1):3133–3181, 2014.

[14] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences, 89(22):10915–10919, 1992.

[15] Thomas R Ioerger, Larry A Rendell, and Shankar Subramaniam. Searching for representations to improve protein sequence fold-class prediction. Machine Learning, 21(1-2):151–175, 1995.

[16] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai, volume 14, pages 1137–1145, 1995.

[17] Max Kuhn. Building predictive models in r using the caret package. Journal of Statistical Software, 28(5):1–26, 2008.

[18] Max Kuhn and Kjell Johnson. Applied predictive modeling. Springer, 2013.

[19] Manish Kumar, Michael M Gromiha, and Gajendra PS Raghava. Identification of dna-binding proteins using support vector machines and evolutionary profiles. BMC bioinformatics, 8(1):463, 2007.

[20] Jie Li and Patrice Koehl. 3d representations of amino acidsâĂŤapplications to protein sequence comparison and classification. Computational and structural biotechnology journal, 11(18):47–58, 2014.

[21] Harvey F Lodish, Arnold Berk, S Lawrence Zipursky, Paul Matsudaira, David Baltimore, James Darnell, et al. Molecular cell biology, volume 4. Citeseer, 2000.

[22] Charles E Metz. Basic principles of roc analysis. In Seminars in nuclear medicine, volume 8, pages 283–298. Elsevier, 1978.

[23] Hiroshi Nakashima, Ken Nishikawa, and OOI Tatsuo. The folding type of a protein is relevant to the amino acid composition. Journal of biochemistry, 99(1):153–162, 1986.

[24] Loris Nanni and Alessandra Lumini. A new encoding technique for peptide classification. Expert Systems with Applications, 38(4):3185–3191, 2011.

[25] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. An empirical study of different approaches for protein classification. The Scientific World Journal, 2014, 2014.

[26] Andrew Y Ng. Preventing" overfitting" of cross-validation data. In ICML, volume 97, pages 245–253, 1997.

[27] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[28] R Bharat Rao, Glenn Fung, and Romer Rosales. On the dangers of cross-validation. an experimental evaluation. In SDM, pages 588–596. SIAM, 2008.

[29] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. Kernel Methods in Computational Biology, pages 35–70, 2004.