A CLUSTERING METHOD FOR THE PROBLEM OF PROTEIN SUBCELLULAR LOCALIZATION

PERİT BEZEK

DECEMBER 2006

A CLUSTERING METHOD FOR THE PROBLEM OF PROTEIN SUBCELLULAR LOCALIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

PERİT BEZEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2006

Approval of the Graduate School of Natural And Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master Of Science.

Prof. Dr. Ayşe KİPER
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master Of Science.

Prof. Dr. Volkan ATALAY
Supervisor

Examining Committee Members

Assoc. Prof. Dr. İsmail Hakkı Toroslu	(METU,CENG)	_____
Prof. Dr. Volkan Atalay	(METU,CENG)	_____
Asist. Prof. Dr. Erkan Mumcuoğlu	(METU,II)	_____
Dr. Tolga Can	(METU,CENG)	_____
Ömer Sinan Saraç (M.Sc.)	(METU,CENG)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Lastname : Perit BEZEK

Signature :

Abstract

A CLUSTERING METHOD FOR THE PROBLEM OF PROTEIN SUBCELLULAR LOCALIZATION

Bezek, Perit
M.Sc., Department of Computer Engineering
Supervisor: Prof. Dr. Volkan ATALAY

December 2006, 68 pages

In this study, the focus is on predicting the subcellular localization of a protein, since subcellular localization is helpful in understanding a protein's functions. Function of a protein may be estimated from its sequence. Motifs or conserved subsequences are strong indicators of function. In a given sample set of protein sequences known to perform the same function, a certain subsequence or group of subsequences should be common; that is, occurrence (frequency) of common subsequences should be high.

Our idea is to find the common subsequences through clustering and use these common groups (implicit motifs) to classify proteins. To calculate the distance between two subsequences, traditional string edit distance is modified so that only replacement is allowed and the cost of replacement is related to an amino acid substitution matrix. Based on the modified string edit distance, spectral clustering embeds the subsequences into some transformed space for which the clustering problem is expected to become easier to solve. For a given protein sequence, distribution of its subsequences over the clusters is the feature vector which is subsequently fed to a classifier. The most important aspect if this approach is the use of spectral clustering based on modified string edit distance.

Keywords: Protein Classification, Subcellular Localization, Spectral Clustering, String Edit Distance, Implicit Motifs

Öz

PROTEİNLERİN HÜCRE İÇİ YERLEŞİMLERİNİ BULMAK İÇİN BİR KÜMELEME YÖNTEMİ

Bezek, Perit
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Volkan ATALAY

Aralık 2006, 68 sayfa

Bu çalışmanın odak noktası proteinlerin hücre içi yerleşimlerini bulmaktır çünkü hücre içi yerleşim bir proteinin işlevlerini anlamada gayet yardımcı olacak bilgiler içerir. Bir proteinin işlevleri amino asit dizisinden kestirilebilir. Motifler ya da korunan altdiziler güçlü bir şekilde belirli bir işlevin varlığına işaret eder. Aynı işleve sahip olduğu bilinen bir grup protein dizisinde, belirli bir altdizi ya da belirili bir altdizi grubu sıkça rastlanır olmalıdır yani bu altdizi gruplarının görülme sıklığı, frekansı, yüksek olmalıdır.

Bizim fikrimiz bu ortak altdizileri öbekleme yöntemi ile bulmak ve onları (implicit motifs) proteinleri sınıflandırmak için kullanmaktır. İki altdizi arasındaki mesafeyi hesaplamak için geleneksel metin düzenleme uzaklığı, sadece harflerin değiştirilmesine izin verecek şekilde uyarlanmış ve değişirme masrafı da bir amino asit benzerlik matrisine bağlı olacak hale getirilmiştir. Tayfsal öbekleme, bu yeni metin düzenleme uzaklığını baz alarak altdizileri başka bir uzaya göndermektedir; böylece kümeleme problemi daha kolay çözülür hale gelmektedir. Verilen bir protein dizisi için altdizilerinin öbeklere göre dağılımı bir sınıflandırıcıya verilecek olan özellik vektörünü oluşturmaktadır. Bu yaklaşımın en önemli kısmı metin düzenleme uzaklığı üzerine kurulan tayfsal öbeklemedir.

Anahtar Kelimeler: Protein Sınıflandırma, Hücre İçi Yerleşim, Tayfsal Öbekleme, Metin Düzeneleme Uzaklığı, Örtük Motifler

To my father,
who encouraged me a lot for this thesis
but could not see it accomplished

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Dr. Volkan ATALAY, without whose guidance and support this work could not be accomplished.

I deeply thank the members of the Department of Computer Engineering and my friends for their suggestions during the period of writing the thesis.

I deeply thank my family for their understanding and their support.

Table of Contents

APPENDICES

List of Algorithms

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem

When considered from its functional point of view, an eukaryotic cell has different compartments called subcellular localizations. Each of these compartments host different cellular processes and proteins functionally linked with those processes target that compartment in the cell. In most of the cases, this compartment, the subcellular localization of the protein, is determined by a short amino acid sequence segment of the whole protein sequence. From this point forward a short amino acid sequence segment will be called a subsequence. A subsequence determining the subcellular localization is called protein sorting signal.

Studying the subcellular localization of proteins, therefore, is very important in understanding their functions and their role in the lifecycle of a cell. Since certain cell functions occur in certain subcellular localizations, estimating the subcellular localization of a newly discovered or unknown protein will yield information about its functions.

Because of these facts an algorithm, an automated system to compute the subcellular localization of a protein is highly needed in order to classify newly discovered proteins or manufactured proteins.

1.2 Motivation And Contribution

The functions of protein a may be estimated from its sequence. Protein sequences with same functionality, or same subcellular localization in this case, have common subsequences or have common subsequence groups. Our assumption is that, in a given set of protein sequences with the same subcellular localization, subsequences belonging to certain common subsequence groups have a high number of occurrence. In other words, for a given subcellular localization, the protein sequences targeting it will have some common subsequence groups with high frequencies.

The ideas in this work originates from the above assumption. The method proposed in this thesis work is to compute these common groups using clustering, namely spectral clustering. This study features the use of string edit distance to compute the similarity of subsequences and spectral clustering to obtain subsequence clusters which are the most important aspects of this work. The aim of clustering is actually to find the mentioned subsequence groups. The actual classification of protein sequences is carried on features based on these groups.

There are four kinds of subcellular localization targets, or classes from computational point of view, in this system:

- Proteins targeting Endoplasmic Reticulum - ER
- Proteins targeting Mitochondria
- Proteins targeting Cytoplasm
- Proteins targeting Nucleus

This problem, computationally, is actually an n-class classification problem. The classification is based on the feature vectors which are mappings of protein sequences to a feature space. The mapping is based on some relations between the protein sequence and the groups. The idea is to train the system with protein sequences

of known localizations which will allow the system to learn possible feature space mappings for each particular compartment.

1.3 Organization

This thesis is organized in five main chapters, including this introduction chapter as the first chapter. In the second chapter a background information is given. To let the reader understand the problem domain, first biological background is presented which is succeeded by a survey on the bioinformatics studies on the subject, ended by background information on major algorithms used in this work. In the third chapter the proposed method is presented. First, the suggested method is presented in an overall fashion. Then main parts in running and training phases are described in detail. The results obtained by the method on the data set are presented on the fourth chapter together with a discussion. The last chapter contains the conclusion and the future work.

Chapter 2

Biological and Computational Preliminaries

2.1 Biological Background

2.1.1 Cellular Nucleic Acids

DNA

All cells, including some viruses, contains Deoxyribo Nucleic Acid (DNA). DNA is a long chain of nucleotides. It contains the genetic information of the organism and this information is used for the biological development of that organism. Since DNA contains genetic information, it is the basic structure for inheritance, that is, DNA is responsible for passing the inherited traits throughout generations.

DNA is in the shape of a double helix organized as strands. Each strand of this double helix is called a *nucleotide* which is the building blocks of the DNA. There are four different types of nucleotides: Adenine (A), Cytosine (C), Guanine(G) and Thymine (T).

Each type of strand can have exactly one type of strand as its complement. This is due to the fact that each nucleotide type can bond with only a single nucleotide

type: A bonds with T, C bonds with G. Two nucleotides paired together are called a base pair. By using one side of the helix, the other side can be computed.

RNA

Another type of nucleic acid found in cells is the Ribo Nucleic Acid (RNA). RNA, like DNA, is a nucleic acid polymer consisting of nucleotide monomers. The main differences between RNA and DNA are:

1. RNA has ribose whereas DNA contains deoxyribose.
2. Instead of the double strands in DNA, RNA has a chain of single strands.
3. RNA has Uracil instead of Thymine.
4. Compared to DNA, RNA has a smaller size and internal bond strengths are weaker.

The Central Dogma

As the focus of this study is on proteins, it has to be stated that there is an important relationship between nucleic acids and proteins. Actually, this relationship is so important that it is called *The Central Dogma*. The central dogma is as follows,

> DNA molecules contain information about how to create proteins; this information is transcribed into RNA molecules, which, in turn, direct chemical machinery which translates the nucleic acid message into a protein.

2.1.2 Cells

Cells are the building blocks of life. They are the structural and functional units of living organisms. Simple organisms like bacteria consist of a single cell whereas complex organisms like humans consist of many many cells. For example human body

roughly contains 100 trillion cells. Cells host all the vital functions of a living organism. They also contain the genetic information of that organism. This information is used for controlling and regulating the cell functions and it is transmitted to next generation of cells.

Cells can be divided into two major groups: prokaryotic cells and eukaryotic cells. The major difference between prokaryotic and eukaryotic cells is that eukaryotic cells have a nuclear membrane, which envelopes the DNA. Prokaryotic cells also lack some other membrane bounded compartments like mitochondria or Golgi apparatus. Eukaryotic cells are 10 times larger than prokaryotic cells on the average. Prokaryotic cells are usually observed in single-cellular organisms whereas eukaryotic cells are seen in multi-cellular organisms.

All cells, whether prokaryotic or eukaryotic, have a membrane, which covers the cell and separates it from its surroundings maintaining its integrity. Inside this membrane there is the cytoplasm, a salty plasma which takes up most of the cell's volume. Other structures of the cell float inside this cytoplasm.

Cells also contain many small structures to carry out different activities. These activities are necessary for a cell to carry out its existence. They are the vital part of a cell's biological development. These small biological structures are called organelles. Some of the important organelles are:

- **Nucleus:** The cell nucleus hosts the cell's DNA, that is the genetic information of the cell. Most of the DNA replication and RNA synthesis processes occur in the nucleus. A membrane separates the nucleus from cytoplasm and gives it a spherical shape.

- **Mitochondria:** Mitochondria can be found in different shapes and sizes in the cytoplasm of a cell. They play an important role in the generation energy for the cell. They have their own DNA separate from the DNA in the cell nucleus. Hence they have the ability of self-replication.

- **Endoplasmic Reticulum (ER):** ER is the transportation mechanism of the cell. It transfers molecules to their targets which may be an area for processing the molecule or an area for the molecule to collaborate in a certain cellular function.

- **Ribosomes:** Ribosomes are the organelles where synthesis of protein molecules occur. By using the genetic code in an RNA, they create a sequence of amino acids to make up a protein molecule. Ribosome, itself, is a protein complex, a large protein structure, which is a bit different from other organalles.

- **Lysosomes and Peroxisomes:** Lysosomes and peroxisomes contain destructive enzymes for degrading proteins, nucleic acids, and polysaccharides. Therefore, they are usually called as the garbage disposal system of a cell.

- **Centrosomes:** Centrosome directs the transport through the ER and the Golgi apparatus.

- **Vacuoles** Vacuoles can be considered as silos of the cell. They store food and waste.

All these organelles actually correspond to different subcellular localization targets. Proteins, and other molecules that take part in a specific cellular activity target the related organelle. For example, proteins, or enzymes related to the production of other protein molecules target the ribosomes, whereas proteins related to the production of energy target the mitochondria. Since each organelle hosts different cellular tasks, finding, or computing, the target organelle for a certain protein molecule gives information about its contribution to the cellular life, thus its functions.

2.1.3 Proteins

Proteins are large organic molecules that play many important roles in living organisms. Proteins take part in nearly all the processes that occur in cells. They may

act as enzymes which catalyze biochemical reactions in cells. They may be used as building blocks in the cytoskeleton of a cell. They also take part in cellular activities like cell signaling, immune responses, cell adhesion, and the cell cycle.

Proteins are actually sequences of amino acid molecules where the carboxyl part of one amino acid has a bond with the amine nitrogen of the other amino acid. These bonds between the consecutive amino acids of a protein molecule is called a peptide bond. The amino acid sequence of a protein molecule is called its *primary structure*.

As the central dogma states, the amino acid sequence of a protein molecule is determined by the genetic information of the cell that hosts the chemical production process. Parts of the genetic information which resides in the DNA of the cell, is transfered using RNA molecules. The process of copying of genetic information stored in the DNA to RNA is called *transcription*. The information in RNA is then used to form the amino acid sequence of a protein molecule. The process of forming the amino acid sequence using information in RNA is called *translation*. The translation of RNA into a sequence of amino acids takes place in ribosomes of the cells. It must be noted that different parts of the DNA are used for different types of proteins, so the production information for a specific type of protein molecule is stored in a specific part of the DNA.

The amino acid sequence of a protein determines its conformation. Therefore it determines its shape and function. The exact process of this determination, however, is currently unknown and it is one of the major unsolved problems in molecular biology[21].

The overall conformation of the whole protein molecule is called its *fold*, or *tertiary structure*. The process of a protein molecule's gaining its overall shape, which is mainly determined by its amino acid sequence, is called folding.

Folding is quite an important process because the shape of a protein has important aspects in its functions. For example, many enzymes have specific shapes for their substrates. These shapes are based on the shape of their respective substrates and

allow easy interaction of the enzyme - substrate combination.

From a computer scientist's point of view, folding can be considered as determining the coordinate vectors for each element of a long sequence where elements are members of a 20-letter alphabet. This is what actually happens in folding. Ribosomes first translate the RNA into an amino acid sequence. Then by chemical processes, this amino acid sequence folds to form the final conformation of the actual protein molecule. This process takes only a few seconds to complete. However, it is quite a difficult task for us to compute the folding of a protein molecule from its amino acid sequence.

There are also local shapes for parts of a protein molecule. There are particular shapes seen in many protein repeatedly for segments of the amino acid sequence where the length of these sequences are a few dozen amino acids. These local shapes are called the *secondary structure*. α-helices and β-sheets are the major secondary structures. Secondary structures are important in the sense that certain combinations of secondary structures cause proteins to act in certain ways. Putting it in the other way around, in proteins behaving in a particular way, certain secondary structures have been observed. For example, two α-helices linked by a turn with an approximately 60 angle have been observed in a variety of proteins that bind to DNA[21]. It can be said that the existence of certain secondary structures causes proteins to target certain localization in cells.

2.2 Computational Studies on Subcellular Localization Prediction

Protein motifs are subsequences in an amino acid sequence which are specifically related to a certain biological function. Determining these protein motifs is a key element in accurate protein sequence annotation. In general, computational motif discovery tools [4, 6], focus on the explicit search and identification of motifs. There

are three major approaches to identify motifs [2, 4, 5, 8, 19]:

- Deterministic patterns (PROSITE, PRINTS)
- Profiles (BLOCKS, PROSITE, MEME)
- Probabilistic patterns with hidden Markov models (PFAM)

Finding the subcellular localization of protein gives information on its function. Therefore analysis of subcellular localization for new proteins is of high importance for understanding their functions. There are many different studies on subcellular localization prediction. Some are based on the primary structure of the protein molecule, namely its amino acid sequence, whereas others concantrate on the tertiary structure of the protein, namely its three dimensional structure or its fold [11, 13, 27, 29]. Traditional subcellular localization predictors like TargetP, SignalP and NNPSL use machine learning algorithms to detect the presence of signal peptide leavage sites on the amino acid sequence of protein molecules.

TargetP [14], a well-known system for subcellular localization prediction, predicts the subcellular location of eukaryotic proteins. The location assignment is based on the predicted presence of any of the N-terminal presequences: chloroplast transit peptide, mitochondrial targeting peptide or secretory pathway signal peptide. It can also predict a potential cleavage site for sequences predicted to contain an N-terminal presequence

New methods emerge using extensive biological knowledge together with machine learning algorithms. P2SL, LOC3D, PA-SUB, PSORT-B and SMART [1, 14, 17, 23, 27, 28] are some of these methods and they have better prediction rates than traditional methods.

P2SL [1], is one of the new prediction methods. It models targeting-signal by the distribution of subsequence occurrences using self organizing maps. It determines the class boundaries using a set of support vector machines. P2SL has a high prediction

rate. It also gives the distribution potential of proteins among different localization classes.

2.3 Spectral Clustering

Spectral clustering refers to a class of techniques which rely on the partioning the weighted graph constructed from a given set of data in order to specify the clusters. Each node in the graph corresponds to a data point while the weight on each edge corresponds to the similarity between the two nodes it connects. A clustering then represents a multiway cut in the graph. In the sequel, we follow the notation of von Luxburg [36].

2.3.1 Preliminaries

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, \ldots, v_n\}$. Assume that each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$; that is the graph G is weighted. Let $W = (w_{ij})_{i,j=1,\ldots,n}$ represent the weighted adjacency matrix of the graph. If $w_{ij} = 0$, then the vertices v_i and v_j are not connected. Since G is undirected, $w_{ij} = w_{ji}$. The degree d_i of a vertex $v_i \in V$ is defined as follows.

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{2.1}$$

We can then define D, degree matrix as the diagonal matrix with $d_1, \ldots, d_n$ on the diagonal.

Given a subset of vertices $A \subset V$, we denote its complement by $\bar{A}$. There are two ways of measuring the size of a subset: $A \subset V$: $|A|$ and $vol(A) = \sum_{i \in A} d_i$. Intuitively, $|A|$ measures the size of A by its number of vertices while $vol(A)$ measures the size of A by the weights of its edges. If $A_i \cap A_j = \emptyset$ and $A_1 \cup \ldots \cup A_k = V$, then the sets $A_1, \ldots, A_k$ form a partition of the graph.

2.3.2 Clustering as Graph Partitioning

Clustering is to group data according to their similarities. When data is represented in the form of a similarity graph, clustering can be restated as follows: find a partition of the graph such that the edges between different subsets have a very low weight (which means that data in different clusters are dissimilar from each other) and the edges within a subset have high weight (which means that data within the same cluster are similar to each other). In this sense, a graph G can be partitioned into two disjoint sets, A and B by simply removing edges connecting the two parts. The similarity of these two parts can be computed as the total weight of the edges that has to be removed. Formally,

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij} \tag{2.2}$$

When this cut value is minimized, optimal bipartitioning of graph is obtained. There are several efficient algorithms for solving minimum-cut problem [34]. However, minimum-cut does not take into consideration the size of the subgraphs that it tries to separate. In multiway cut, minimum cut is

$$cut(A_1, \ldots, A_k) = \sum_{i=1}^{k} cut(A_i, \bar{A}_i) \tag{2.3}$$

and it suffers from the same problem. In order to overcome this problem, we may require the sets $A_1, \ldots, A_k$ to be somewhat large. In the literature, RatioCut [18] and the normalized cut (NCut) [31] are the most popular cut definitions that incorporate this requirement.

$$RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|} \tag{2.4}$$

$$NCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)} \tag{2.5}$$

RatioCut measures the size of a subset A of a graph by its number of vertices $|A|$ and Ncut measures it by the weights of its edges $vol(A)$. One disadvantage of the size requirement is that the solution becomes NP-hard [37].

2.3.3 Laplacian of a Graph and Solution to Graph Partitioning

Partitioning of a given graph can be performed through the analysis of spectrum of its weighted adjacency matrix W. Spectrum of a matrix is its eigenvectors, ordered by the magnitude of their corresponding eigenvalues. Eigenvalues and eigenvectors of a matrix provide global information about its structure.

Laplacian of the graph matrices are the main tools for spectral clustering and spectral graph theory is a field studying these matrices [9]. Note that W represent the weighted adjacency matrix and D is the degree matrix. Since we work on undirected graphs, W is a symmetric matrix. The unnormalized graph Laplacian matrix is then defined as follows [25, 26].

$$L = D - W$$

The following proposition summarizes the most important facts needed for spectral clustering [36].

Proposition 1 (Properties of L) The matrix L satisfies the following properties:

1. For every vector $f \in \mathbb{R}^n$ we have

$$f'Lf = \frac{1}{2}\sum_{i,j}^{n} w_{ij}(f_i - f_j)^2$$

2. L is symmetric and positive semi-definite.

3. The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector 1.

4. L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

A normalized graph Laplacian is defined as a symmetric matrix.

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \tag{2.6}$$

Second-smallest eigenvalue of the Laplacian of graph has been associated with the connectivity of the graph [15]. One can, in fact define a semi-optimal cut by using second eigenvector of a graph's Laplacian. By this way, graph partitioning problem which is NP-hard is relaxed and it can be shown that cuts based on the second eigenvector give a guaranteed approximation to the optimal cut. More specifically, the second smallest eigenvector of the generalized eigensystem $(D - W)y = \lambda Dy$ is the real valued solution to the normalized cut problem [15, 31, 32].

2.3.4 Spectral Clustering Methods

There are several spectral clustering algorithms which are empirically very successful. They differ in

- which matrix they use (normalized and unnormalized graph Laplacian)
- which eigenvectors they use, and
- how they use the eigenvectors exactly.

[38], [35] and [36] present overviews of some of the algorithms. The basic spectral clustering algorithm [16] can be given as follows:

Algorithm 1 Basic Spectral Clustering

1: Build the affinity matrix.
2: Determine the dominant eigenvalues and eigenvectors of the matrix.
3: Use these to compute the clustering.

Spectral algorithms may be one of the two general types.

1. *Recursive Algorithms* recursively split data points into two using only one eigenvector at a time until K partitions are obtained. A recursive spectral clustering

algorithm is a solution to the optimization problem of calculating the minimum normalized cut of a graph by using the second smallest eigenvector. The first eigenvalue being 0, is the trivial solution. In recursive spectral clustering algorithms, data is partitioned into two groups using, for example the second eigenvector and this recursive partitioning continues until the desired number of clusters is obtained.

2. *Multiway Algorithms* directly split data points into K clusters based on the largest K eigenvectors of the normalized graph Laplacian matrix. Multiway algorithms use these eigenvectors as some sort of basis vectors and project data points in a new K dimensional space. This is actually called spectral embedding where data points are arranged such that simple clustering algorithms yield successful results.

Among spectral clustering algorithms, the following four algorithms are very popular.

- Meila-Shi (Multicut-MS) algorithm [24],
- Ng, Jordan, Weiss (NJW) algorithm [30],
- Shi and Malik (SM) algorithm [31],
- Kannan, Vempala and Vetta (KVV) algorithm [22].

SM and KVV are recursive algorithms while Multicut-MS and NJW are multiway algorithms.

Some comparisons based on experimental results show that the multiway algorithms work better when data contains little or no noise, but the recursive algorithms are more noise tolerant and produce better results under high noise. Besides, if the choice of K is larger than the actual number of clusters, multiway algorithms perform worse than their recursive counterparts. However, under normal conditions the NJW algorithm seems to be a slightly better choice [35].

Comparing both approaches, [12] and [38] came to the conclusion that the normalized version should be preferred.

2.3.5 Building the Affinity Matrix

The graph is constructed by connecting each point to its k nearest neighbors in the space, or by connecting each point to all neighbors within distance ϵ. We assign symmetric non-negative edge weights. For numerical data, a convenient weighting function is the Gaussian kernel. w_{ij}: $e^{\beta||x_i - x_j||}$ if i and j are connected and 0 otherwise.

A slight advantage of the Gaussian function is that it results in a positive definite affinity matrix-a kernel matrix simplifying the analysis of eigenvalues. The parameter σ is a user-defined value, referred to as the kernel width. The choice of a correct width is critical for the performance of the algorithm.

2.3.6 Ng, Jordan, Weiss Spectral Clustering Algorithm

Since NJW algorithm is the spectral clustering of choice in this work, it is necessary to give further information about this algorithm which can be found in [30]:

Assume we are given a set of points $S = \{s_1, ..., s_n\} \subset \mathbb{R}^m$ which we want to partition into k clusters, the algorithm is given in Algorithm 2.

Algorithm 2 NJW Spectral Clustering

1: $A_{ij} \leftarrow e^{-\frac{\|s_i - s_j\|^2}{2\sigma^2}}$ $\{A \in \mathbb{R}^{n \times n}$ and is called the affinity matrix$\}$
2: $D_{ii} \leftarrow \Sigma_j A_{ij}$ $\{D$ is the a diagonal matrix called volume matrix$\}$
3: $L \leftarrow D^{-1/2} A D^{-1/2}$ $\{L$ is the Laplacian matrix$\}$
4: find the largest k eigenvectors $x_1, ..., x_k$ of L
5: $X \leftarrow [x_1 x_2 ... x_k]$ $\{X \in \mathbb{R}^{n \times k}\}$ $\{X$ is formed by stacking the eigenvectors in columns$\}$
6: calculate the matrix Y by normalizing rows of X to unit length $\{$

$$Y_{ij} \leftarrow \frac{X_{ij}}{\sqrt{\Sigma_j X_{ij}^2}}$$

$\}$
7: use K-Means Clustering to partition the rows of Y into k clusters $\{n$ points that are vectors in $\mathbb{R}^k\}$
8: assign the original point i to cluster j if and only if the i^{th} row of Y has been assigned to cluster j.

This algorithm contains sophisticated mathematics in background (like *Matrix Perturbation Theory*[33]) and to gain full understanding of the method one has to study each step of the algorithm to see the reasons and methods used to manipulate the data. Here is a brief explanation of the steps in the sense that they are used for clustering.

1. Simply form a similarity matrix based on the chosen distance metric.

2. Calculate the total similarity of each point to be used in next step for normalization.

3. This is one of the crucial steps. By this multiplication the value L_{ij} is now the normalized version of W_{ij}; that is, it is a percentage value obtained considering both i and j.

4. Select k largest eigenvectors to represent the data. The idea behind this selection originate from the fact that in ideal case the eigenvectors correspond to maximal eigenvectors of the k clusters respectively[30]. At this step the rows of X represent the points, that is their coordinates using the selected k

eigenvectors to be a basis of the new transformed space.

5. The normalization in this step can be considered as a means of projection. As mentioned above what is actually done in this method is mapping the original data to a $k-sphere$ which is also called *spectral embedding*. This normalization results in setting the length of rows of X, the mapped points, to unit length which corresponds to projecting them to the surface of a $k-sphere$ with unit radius.

6. If the parameter σ is adjusted well, the resulting mapping is expected to map the original data to convex disjoint regions on the surface of the infamous $k-sphere$. Now it is rather straightforward for K-means to cluster these points. Of course other clustering algorithms can be utilized. However a simple algorithm is favored because much of the work has been done by the *spectral embedding*.

7. This final step is just labeling the original data according to the labels of mapped data.

As seen above the crucial steps are actually simple normalization steps. They result in grouping and projection of data. These two points are the key components of this method.

2.4 Support Vector Machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Their common factor is the use of a technique known as the "kernel trick" to apply linear classification techniques to non-linear classification problems. We use SVM as a 2-class classifier, so the details will be given for binary classification.

In a 2-class classification problem we have feature vectors $x_i \in \mathbb{R}^m$ where $1 \leqslant$

$i \leqslant N$ and corresponding class labels $c_i \in \{+1, -1\}$ where $+1$ indicates one class and -1 indicates the other class. SVM finds a boundary separating the vectors in different classes. It maps x_i to a higher dimensional space $\mathbb{H}$ using a mapping Φ where $\Phi(x_i) \in \mathbb{H}$. After the mapping, the mapped feature vectors can be separated by a hyperplane in $\mathbb{H}$. SVM finds a hyperplane separating the classes where the hyperplane maximizes the distance between itself and the nearest vectors to it from each class. The distance between the hyperplane and of the nearest vectors from each class to the hyperplane itself is called the *margin*. So SVM finds the hyperplane with the maximum margin. This hyperplane is called the Optimal Separating Hyperplane (OSH).

The mapping in SVM is done via a kernel function $K(x_i, x_j)$. This kernel function defines an inner product in the space $\mathbb{H}$. The decision function of SVM is

$$f(x) = sgn(\sum_{i=1}^{N} c_i \alpha_i \cdot K(x, x_i) + b) \tag{2.7}$$

where α_i are obtained solving the following equation:

$$\text{Maximize} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \cdot c_i c_j \cdot K(x_i, x_j) \tag{2.8}$$

$$\text{subject to} \quad 0 \leqslant \alpha_i \leqslant C, \quad \sum_{i=1}^{N} \alpha_i c_i$$

In the equation 2.8, C is a regularization parameter. It controls the tradeoff between margin and misclassication error. The x_j with corresponding $\alpha_i > 0$ are called the *Support Vectors*. They are the closest vectors to the OSH from each class.

There are several different types of kernel functions and their choice affects the performance of the SVM on a particular set of vector.

- *Polynomial kernel function* which has a structure of $K(x_i, x_j) = (x_i \bullet x_j + 1)^d$

where d defines the degree of the polynomial

- *Radial Basis Function (RBF)* which has a structure of $K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$ with the parameter γ

are two popular and widely used kernel functions. A different kernel function can be designed for a particular problem. This is, however, preferred when existing kernel functions are not sufficient for the problem. For a given problem and data set, one has to choose a kernel function and the regularization parameter C to generate the SVM model.

Chapter 3

System and Modules

3.1 System

We assume that a set of protein sequences with the same subcellular localization have common subsequences or groups of similar subsequences. The occurrence of these common subsequence groups should be high for protein sequences targeted to the same subcellular localization. The main idea of this study is to use these common subsequence groups to predict subcellular localizations for given proteins. The system for the prediction of subcellular localization is shown in Figure 3.1 and it is composed of modules whose outputs are binary decisions. Binary decision is on whether input protein sequence is labeled to one of two classes. In our system, in order to cover different combinations of pairs of 4 classes, 6 binary modules are required. Therefore, the system consists of a set of six modules where each module is responsible for a pair of classes, and a decision logic which is required to achieve a final result among six decisions. Current implementation of decision logic is based on majority voting.

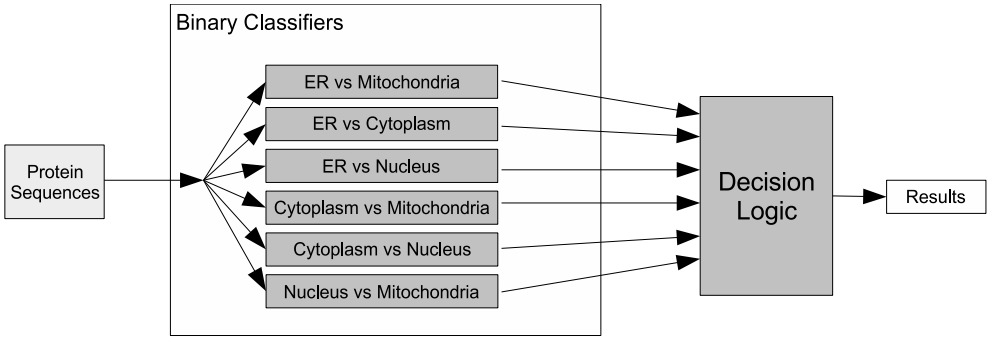

Figure 3.1: Overview of the system.

Binary classifiers form the very core of the system. A binary classifier takes a protein sequence as input and outputs the label of that sequence. Binary classifiers consist of several submodules. The main submodules are *Feature Extractor* and *Support Vector Machine*. As the names suggest, feature extractor extracts features from the protein sequence generating a feature vector and the support vector machine classifies this vector labeling it with one of the two labels this classifier is trained for. A feature extractor also contains submodules which are *Sequence Decomposer*, *Quantizer* and *Feature Vector Generator*. This general structure is shown in Figure 3.2.

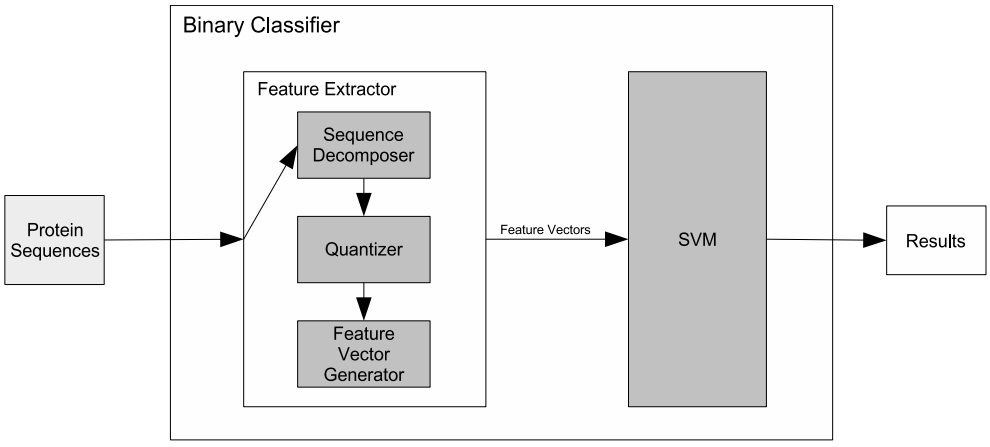

Figure 3.2: Single binary classifier

Leaving the details of the modules to the following sections, a single binary classi-

fier decomposes a protein sequence into its subsequences, extracts features by feature vector generator using subsequences and the quantizer, and generates the feature vector representation of the protein sequence. It, then, classifies the feature vector by the support vector machine.

Each binary classifier has to be trained first so that they will learn how to separate members of one class from the other. The training phase is the phase where the quantizer module in the feature extractor is filled with appropriate data and the support vector machine learns the boundary separating the classes.

The training phase basically consists of two main parts, the clustering part which yields a quantizer and the support vector machine training. These parts are shown in Figure 3.3 and Figure 3.4.

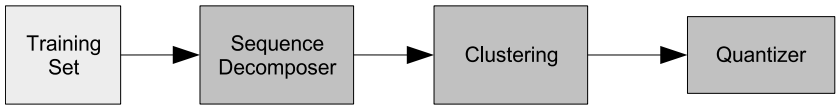

Figure 3.3: Generating a quantizer by clustering

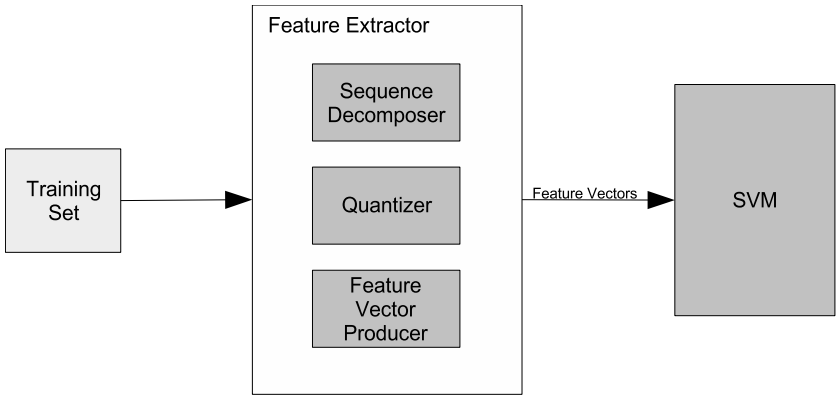

Figure 3.4: Training a classifier

The basic flow of the training phase can be summarized as follows:

Algorithm 3 Binary Classifier Training

1: read the training protein sequence.
2: decompose protein sequences into subsequences.
3: cluster subsequences using spectral clustering and string edit distance.
4: construct a quantizer using the clusters.
5: construct a feature extractor by combining the sequence decomposer, the quantizer, and the feature vector generator.
6: generate feature vectors.
7: train the support vector machine.

We will now give detailed information on the purposes and working details of the modules our system is composed of.

3.2 Modules

A module consists of two main phases: training and test. In the training phase, the input is a training data set P that is composed of the protein sequences p_i each of which belongs to one of the two classes, c_1 or c_2. The output of training phase is groups of clustered subsequences and a trained binary classifier. The following steps are carried out in the training phase.

1. For each sequence p_i in the training set P, decompose sequences into subsequences (P' denotes the whole set of subsequences of the sequences in the training set)

2. Find subsequence groups, $\lambda_1, \ldots, \lambda_k$ where $\lambda_1 \cup \ldots \cup \lambda_k = P'$ based on clustering.

3. For each sequence p_i, generate a feature vector x_i by evaluating a membership function for the subsequences over subsequence groups $\lambda_1, \ldots, \lambda_k$ (Form a training set T for the classifier where each element is a tuple: $t_i \in T, t_i = (x_i, c_i)$ feature vector and its class)

4. Train the classifier with the training set T.

In order to obtain common subsequence groups, spectral clustering based on the replacement argument of string edit distance is applied. String edit distance measures basically the distance between two subsequences. A membership function based on the relation between a protein subsequence and subsequence groups is defined to represent a protein sequence as a feature vector. Feature vector along with the label of protein sequence is subsequently fed to a support vector machine for training purposes.

The following steps are pursued in order to predict the class of a given sequence p_i in the test phase.

1. Decompose the input sequence p_i into subsequences.
2. Generate a feature vector x_i by evaluating a membership function for the subsequences over subsequence groups $\lambda_1, \ldots, \lambda_k$
3. Feed this feature vector x_i to the already trained classifier to get a decision.

3.2.1 Decomposition of a Sequence into Subsequences

A sequence is first decomposed into subsequences. Decomposition is determined by the following three parameters.

- L (total length): This is the number of amino acids considered for extraction purposes from the beginning of the sequence. This can be interpreted as area of interest.
- κ (window size): This is the number of amino acids in a single subsequence extracted from the area of interest in protein. This can be considered as a logical unit which reflects sufficient biological information which affects subcellular localization of the protein.
- t (slide amount): This is the number amino acids that extractor will advance to extract next subsequence. If this number is smaller than window size, sub-

sequences have intersecting amino acids; and if this number is 1 then all subsequences are extracted.

Assume MKTLALFLVLVCVLGLVQSWEWPWNRKPTKFPIPSPNPRDK is given as the amino acid sequence of the protein whose subsequences are needed to be decomposed. Assume $L = 30$, $\kappa = 5$ and $t = 1$ are given as decomposition parameters. Then the subsequences obtained would be as follows:

MKTLA

KTLAL

TLALF

...

WNRKP

NRKPT

RKPTK

3.2.2 Distance Between Two Subsequences

In other to establish a distance metric, string edit distance together with an amino acid similarity index is used. String edit distance is a distance metric [10] which is the distance between two strings given by the number of operations for converting one string to the other. This greatly applies to our case since all needed is the similarity of two subsequences which is actually the *biological cost* of replacing a the amino acids in one two obtain the other. In order to assess the biological cost correctly, an amino acid similarity index is used to calculate cost of substitution of an amino acid with another. The cost of the substitution is inversely proportional with the similarity of two amino acids.

In information theory and computer science, the Levenshtein distance or edit distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is one of the following:

- insertion of a single character
- deletion of a single character
- substitution of a single character

In our system, we did not use insertion and deletion operations. For short subsequences like the subsequences of 5 amino acids length that we used in our system, using insertion an deletion operation can cause every subsequence to look similar and this is not desirable in biological sense. However their introduction can be necessary for long subsequences like the subsequences of 10-15 amino acids because for one or two insertions/deletions of amino acids to obtain a subsequence from another one is sensible biologically and moreover because the subsequences are long these operations allow us to align a long part of them. However assigning a cost to these operations with accurate biological meaning is not an easy task. Because of that, we did not use insertions or deletions in our system.

Assume EDDGF and ADCHF are two given subsequences. The similarity index for this example is the *Isomorphicity of Replacements* matrix which given in Appendix B. String edit distance needs the cost of replacing amino acids. This can be considered as the distance between amino acids. The similarity index provided has to be converted so that it reflects distance, not similarity. The converted version is given in Appendix C. We are not using any insertions or deletions. So the operations to convert EDDGF to ADCHF are:

1. EDDGF $\rightarrow$ ADDGF (E $\rightarrow$ A, cost is 97)
2. ADDGF $\rightarrow$ ADCGF (D $\rightarrow$ C, cost is 89)
3. ADCGF $\rightarrow$ ADCHF (G $\rightarrow$ H, cost is 63)

The total cost to convert ADDGF to BDCHF is 249. So the distance between these two subsequences is 249.

3.2.3 Clustering The Subsequences

Clustering of subsequences is simply performed by using NJW spectral clustering algorithm[30]. Similarity (affinity) matrix is constructed from the subsequences of the all proteins in the training set and using the amino acid similarity index based replacement distance. First the amino acid similarity index is converted so that the converted matrix reflects the distance between amino acids. Using string edit distance with the converted matrix, the distance between each subsequence is calculated. Using the Gaussian kernel of the NJW algorithm, these distance values are converted to similarities which form the affinity matrix. The rest of the NJW spectral clustering algorithm is performed on this affinity matrix. At the end of clustering, K clusters and their member subsequences are obtained.

The most time consuming part of the NJW algorithm is the eigenvalue decomposition of the Laplacian matrix. Given an $N \times N$ matrix, the complexity of the eigenvalue decomposition is $\Theta(N^3)$. Clustering 5000 subsequences even in a powerful computational environment takes a long time. In case of doubling the input size to 10000 subsequences, the clustering times is multiplied by 8. If the time required to cluster 5000 subsequences is 9 hours, which is the case in our training environment, then the time required to cluster 10000 subsequences is 72 hours. Because spectral clustering uses the Laplacian matrix for eigenvalue decomposition, it has a memory complexity of $\Theta(N^2)$. Since the Laplacian matrix contains distances, for an input of 5000 subsequences, 5000×5000 floating point numbers are used. If double precision floating points are used, which are 8 bytes each, the required memory is 200000000 bytes which is nearly 190 MBs. Doubling input size means quadrupling the required memory. For 10000 subsequences, the required memory would be 720 MBs. It must be also noted that this memory has to be allocated throughout the entire process.

To decrease the demanding time and memory complexities of the NJW algorithm, a two pass version of the algorithm has been developed. This modified version divides the whole input set into smaller subsets and clusters each subset individually using

NJW algorithm. Representatives from the clusters of these subsets are chosen and clustered once again to obtain final clusters. The following section describes this method in detail.

3.2.4 Two Pass Spectral Clustering

The two pass method for spectral clustering is devised to decrease the time and memory complexities of the original NJW algorithm. This algorithm consists of two main steps:

1. *Divide input into subsets and cluster:* In the first pass, the input is divided into smaller subsets. The sizes of subsets should be small enough so that clustering takes a feasible amount of time and memory. Obtaining too small subsets, however, causes the system to miss the overall structure. Subsets reflecting the general distribution of the input should be preferred. Subsets are clustered in an iterative fashion using the NJW algorithm.

2. *Collect representatives and re-cluster:* In the second pass, a representative from each cluster is collected. The representative of a cluster is the subsequence which is closest to the other subsequences in that cluster. These representatives are then clustered once again and the final clustering is obtained.

The main idea behind this schema can be summarized as follows. For each subset, the clustering yields subsequence groups with common properties. The members selected from these groups represent those common properties as best as possible, with least error in other words. So the basic purpose of the first pass is to obtain representative subsequences for different groups. The second pass clusters these representatives to find the actual clusters.

A key point in this method is establishing the subsets. If the subsets are not formed in a suitable fashion, similar subsequences may fall into different subsets and into totally different clusters, causing the system to miss their common properties.

In order to have good representatives from subset clusters, subsets should be formed to have different groups of similar subsequences. A good initialization should achieve three main goals:

1. Partition the input in a quick fashion.
2. Put close subsequences into the same subset as much as possible.
3. Partition the input data into evenly distributed subsets.

The importance of the speed factor, that is time complexity, can be understood easily. While introducing a method to improve the speed of spectral clustering, the initialization schema should not introduce a slow component.

Putting close subsequences into the same subset has the purpose of having similar subsequences in the same subset so that the clustering algorithm puts them in the same cluster after the first pass. This way close subsequences will be in the same cluster and this will enable the system to pick a representative subsequence representing common properties of those subsequences. Of course if a certain group of close subsequences has a few number of members, they will probably be a part of a larger cluster and their properties will be reflected less which the system should do.

The size constraint on the subsets serves another purpose. One of the major reasons for using subsets is to speed up the process by decreasing input size for the actual spectral clustering. Using evenly sized subsets guarantees a time limit for the operation and eliminates the probability of having a large subset taking too long to be clustered. Distributing the same number of subsequences to each subsets also ensures that every subset is equally important. A subset with more subsequences than other subsets would have the same amount of representatives after the first pass of clustering. This would cause an representation imbalance on the representative subsequences. Solutions to this problem would be complex and would cause radical changes in the nature of the clustering algorithm. Ensuring equal sizes on the subsets eliminates these problems.

The initialization of subsets problem once again becomes a clustering problem with a size bound on clusters. We have to solve a clustering problem to feed its result to the bigger clustering problem. Using spectral clustering to obtain the initial subsets is obviously out of question; because one of the motivations for this method is the huge memory and time requirements of spectral clustering for such a large data set. In order to solve this initialization problem, we have developed a simple clustering algorithm which is similar to hierarchical clustering with single linkage but which is simpler in its nature. The algorithm is designed to meet the three goals. This algorithm can be called *Bounded Hierarchical Clustering*.

Bounded Hierarchical Clustering

This algorithm is a simple clustering algorithm based on the single linkage hierarchical clustering algorithm. It introduces a size limit on clusters and does not permit cluster merging. These are due to the fact that this algorithm is not designed to be a complete clustering method, but rather an initialization scheme

The algorithm works in an iterative fashion. At each iteration a pair of subsequences with the smallest distance is taken where the final cluster of at least one of the subsequences in the pair has not been decided. The algorithm tries to put these subsequences into the same cluster if possible. If both of the subsequences do not belong to a cluster and maximum number of clusters have not been reached, these subsequences are put in a new cluster. If one of the subsequences belongs to a cluster and that cluster has space for more elements, the other subsequence is put in this cluster. Otherwise, there is no way to put these subsequences into the same cluster obeying the size bounds, so this pair is omitted. This way subsequences that are close to each other are put in the same cluster as much as possible.

We will now describe the algorithm formally. The algorithm takes the subsequences and number of clusters as input parameters. The size of a cluster can be $\frac{number\ of\ subsequences}{number\ of\ clusters}$ at maximum. Heaps used in the algorithm keep the distance

values together with the subsequences related to each distance. Functions used in the algorithm are as follows:

- $insert(d, s, H)$: insert distance d to heap H stating that it is related with subsequence s.
- $pop(H)$: returns and removes the smallest distance and its related subsequence from heap H.
- $get(H)$: returns but keeps the smallest distance and its related subsequence from heap H.
- $clusterSize(c)$: returns the number of subsequences in the cluster c.

The algorithm is as follows:

Algorithm 4 Bounded Hierarchical Clustering

```
1: initialize globalHeap
2: currentNumberOfClusters ← 0
3: maxClusterSize ← inputSize/numberOfClusters
4: for i in subsequences do
5:    initialize heap[i]
6:    for j in subsequences do
7:       d ← dist(i, j)
8:       insert(d, j, heap[i])
9:    end for
10:   insert(get(heap[i]), globalHeap)
11: end for
12: repeat
13:   (d, i) ← pop(globalHeap)
14:   if i is not in a cluster then
15:      (d, j) ← pop(heap[i])
16:      if j is not in a cluster then
17:         if currentNumberOfClusters < numberOfClusters then
18:            put i and j in a new cluster
19:            currentNumberOfClusters++
20:         else
21:            insert(get(heap[i]), globalHeap)
22:         end if
23:      else
24:         c ← clusterOf(j)
25:         if clusterSize(c) < maxClusterSize then
26:            add i to cluster c
27:         else
28:            insert(get(heap[i]), globalHeap)
29:         end if
30:      end if
31:   end if
32: until globalHeap is empty
```

The algorithm is based on hierarchical clustering with single linkage. To ensure the size boundary together with the cluster number requirement, there is no cluster merging. Due to the use of heaps to contain the distance values where insertion and removal of a value has the time complexity of $\Theta(logN)$, the time complexity of the algorithm is $\Theta(N^2logN)$. Although the memory complexity of the algorithm is $O(N^2)$, by deallocating the memory areas for the heaps for subsequences whose clusters are determined, the allocated memory for the algorithm drops fast.

Given N subsequences as input, the time complexity analysis is as follows:

- *insert* and *pop* operations for a heap takes $\Theta(logN)$ time.
- *get* operation does not modify the heap, so it takes $\Theta(1)$ time.
- calculating the distance between two subsequences takes $\Theta(1)$ time.
- For each subsequence an initialization process takes place which takes $\Theta(NlogN)$ time:
  - Its distance to all other subsequences are computed and inserted to heap in an iterative fashion.
  - Each iteration takes $\Theta(1 + logN)$ time which is $\Theta(logN)$.
  - All iterations take $\Theta(NlogN)$ time.
  - The smallest distance is obtained from the heap and inserted into global heap taking $\Theta(1 + logN)$ time which is $\Theta(logN)$.
- These are done for all subsequences so initialization takes $\Theta(N^2logN)$ time.
- In a single iteration to decide the cluster of a subsequence:
  - The smallest distance from the global heap is popped in $\Theta(logN)$ time.
  - The smallest distance for the respective subsequence is popped from its distance heap in $\Theta(logN)$ time.
  - If this subsequence cannot be put into a cluster, the smallest distance is popped from its distance heap and is inserted into the global heap, which take $\Theta(logN)$ time each totaling $\Theta(2 \times logN)$ which is $\Theta(logN)$.
  - A single iterations takes $\Theta(3 \times logN)$ which is $\Theta(logN)$.
- In the best case, every subsequence is put into its cluster in the first time which means N iterations taking a total of $\Theta(NlogN)$ time.
- In the worst case, a subsequence has to be re-inserted back to global heap N times. This will result in N^2 iterations taking a total of $\Theta(N^2logN)$ time.

- In the best case, the algorithm takes $\Theta(N^2logN + NlogN)$ time whereas in the worst case it takes $\Theta(2 \times N^2logN)$ where both of them are $\Theta(N^2logN)$.
- The algorithm takes $\Theta(N^2logN)$ to cluster N subsequences.

Initialization of Subsets

Two different approaches are used to split the input to initialize the subsets using bounded hierarchical clustering. These approaches are:

- *Splitting in one pass:* If the size of the input is N and the maximum size of a subset is given to be M, this approach splits the input into $\frac{N}{M}$ subsets where each subset has M elements.
- *Splitting in two passes:* Assume that the size of the input is N, the maximum size of a subset is given to be M and the number clusters is given to be K. This approach first splits the input into K subsets where each subset has $\frac{N}{K}$ elements which is the first pass. Then all subsets are also splitted into $\frac{N}{M}$ subsets with $\frac{M}{K}$ elements in each. To form the i^{th} subset, the i^{th} subset of all subsets are combined into a single subset. These are performed in the second pass. This approach can be summarized as clustering the input into K clusters and then picking subclusters from these to form the real subsets. The idea behind this approach is to find K clusters in a simpler way first so that each subset will contain certain elements form all of those K clusters. We assume that, after the first pass of the spectral clustering we will have representatives from those K clusters which will simulate the behavior of using the normal spectral clustering.

3.2.5 Quantizer

Protein sequences acting in a similar way, having similar functions, similar subcellular localization compartments have common subsequences from common subsequence

groups. A profile actually contains these groups, these clusters. The clusters are formed in training using spectral clustering.

The clusters in a quantizer are used to quantize a subsequence space, giving this module its name. Each cluster in a quantizer represents a certain portion of the subsequence space. Since there are finite number of clusters in a quantizer, the quantizer represents the space in a quantized fashion. A set of subsequences, possibly subsequences obtained by decomposing a protein sequence, can be converted to a feature vector by computing its relation with each portion defined by this quantizer.

The distance metric used in building the clusters is also a part of the quantizer. The metric is used in calculating the distance between subsequences and clusters. The distance of a subsequence to a cluster of subsequences can be defined as

$$d(s, C) = \frac{\Sigma d(s, C_i)}{|C|}$$

where s is the subsequence of interest, C is the cluster of interest, C_i is the i^{th} subsequence in the cluster and $|C|$ is the number of subsequences in the cluster.

Feature vector generators use these distances to understand the relations between subsequences and clusters and reflect these relations to the feature vectors they generate. It can be stated that quantizers form the core of feature vector generation.

3.2.6 Generating Feature Vectors

A feature vector is a mapping of a protein sequence and it is based on the subsequences of the protein sequence and on the clusters obtained in the training phase. Several alternative methods for generating feature vectors are possible:

a. Generate a feature vector of total distance of subsequence to all clusters. This feature vector has a dimension of the number of subsequences in the protein where the i^{th} entry is the sum of the distances of the i^{th} subsequence of the protein to all clusters in the profile.

b. Generate a feature vector of total distance of each cluster to all subsequences. This feature vector has a dimension of the number of clusters in the profile where the i^{th} entry is the sum of the distances of all the subsequences of the protein to the i^{th} cluster in the profile.

c. Generate a feature vector of distances between each cluster and each subsequence. This feature vector has a dimension of the number of subsequences in the protein times the number of the clusters in the profile where the $i \times j^{th}$ entry is the distance of the i^{th} subsequence of the protein to j^{th} cluster in the profile. This producer is not implemented, so its experimental performance is unknown.

d. Generate a feature vector of cluster frequencies. This feature vector has a dimension of the number of clusters in the profile where the i^{th} entry is the number of subsequences of the protein which are closest to the i^{th} cluster in the profile.

The last method computes the frequency of groups and it is the method used throughout this study. The other methods are implemented to observe their experimental performances.

3.2.7 Classification

We used support vector machines in a one-versus-one setting for classification. This setting is more favorable to one-against-all setting [20]. In a one-versus-one setting, we have to have a classifier for every possible different pair of classes. Because of this we have 6 different classifier in our case to cover all possible pairs formed by 4 classes.

The support vector machine is fed by the feature vectors generated by the methods described in previous sections. The support vector machine is trained by using feature vectors obtained from protein sequences in a training set. Each sequence has a known

class and the support vector machine is fed by these feature vector and class pairs. This way the support vector machine finds a boundary with the maximum margin separating the feature vectors which belong to separate classes. When used as a classifier, the support vector machine takes a feature vector of a protein sequence and decides its class by using the previously computed boundary.

Chapter 4

Results and Discussion

4.1 Preliminary Information

Three different groups of runs were performed on the data set. The first two of these runs are made using the normal spectral clustering algorithm whereas the last group is made using the two pass spectral clustering algorithm.

In all tests, 1680 ER targeted, 880 Mitochondria targeted, 1600 Cytoplasmic and 2900 Nuclear proteins are used. The parameters used for sequence decomposition are taken from P2SL System [1]. The amino acid similarity index used for subsequence distance calculation is *Isomorphicity of Replacements* matrix which is given in Appendix B. The results are presented as the percentage of a predicted label to the actual label.

4.2 Normal Method For 4 Classes

The following parameters are used for this run:

- The number of clusters is chosen to be 100.
- The spectral σ is chosen to be 100.

- For the Cytoplasm vs. Mitochondria classifier, L is chosen to be the whole sequence, κ to be 15 and t to be 5.
- For other classifiers L is chosen to be 30, κ to be 15 and t to be 5.
- For the Cytoplasm vs. Mitochondria classifier, 25 proteins from each class are used for clustering, 100 from each are used for SVM training.
- For other classifiers, 100 proteins from each class are used for the whole training.

The training of a single classifier took approximately 9 hours to complete. The results obtained are given in Table 4.1.

Table 4.1: Results for 4 classes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**83.88**	2.46	3.47	10.17
Mito.	7.95	**70.42**	7.95	13.66
Cyt.	6.72	7.15	**33.18**	52.92
Nuc.	3.62	5.69	11.35	**79.31**

4.3 Normal Method For 3 Classes

Since a total of 50 proteins were used for training the Cytoplasm vs. Nucleus classifier, its results were not good enough, We also tested the system for 3 classes; ER, Mitochondria and Other where the Other class means either Cytoplasmic or Nuclear. During tests, all of 1600 Cytoplasmic and 2900 Nuclear proteins are used as the members of the Other class. The following parameters are used for this run:

- The number of clusters is chosen to be 100.
- The spectral σ is chosen to be 100.
- L is chosen to be 30, κ to be 15 and t to be 5.

- 100 proteins from each class are used for training.

The results obtained are given in Table 4.2.

Table 4.2: Results for 3 classes

Actual	Predicted		
	ER	Mito.	Other
ER	**85.36**	1.58	13.06
Mito.	9.61	**66.39**	24.00
Other	7.10	8.69	**84.21**

4.4 Two Pass Method With 4 Classes

We performed a 4-fold cross validation when using the two pass spectral clustering. For each fold, we performed 4 different tests where the difference between the tests are the input splitting methods used for two pass clustering in training. These 4 different tests are:

1. Split the input using the single pass approach.
2. Split the input using the two pass approach.
3. For classifiers related to ER split the input using the single pass approach and for the rest of the classifiers split the input using the two pass approach.
4. For each classifier use the splitting approach that gave the highest success rate in its training.

We observed that classifiers related to ER perform better when the input is splitted using the single pass approach whereas other classifiers perform slight better when the input is splitted using two pass approach. The 3. test is included because of this fact.

The following parameters are used in all parameters:

- The number of clusters is chosen to be 100.
- The spectral σ is chosen to be 100.
- For the Cytoplasm vs. Mitochondria classifier, L is chosen to be the whole sequence, κ to be 15 and t to be 5.
- For other classifiers L is chosen to be 30, κ to be 15 and t to be 5.
- For all classifiers 200 proteins from each class are used for training:
  - The first 5000 subsequences from each class totaling 10000 subsequences are used for clustering and generating the quantizer.
  - All 400 proteins are used to train the support vector machine.
- The subset sizes are limited to be 1000 subsequences.

The training of a single classifier took approximately 50 minutes to complete.

4.4.1 First Fold

The results obtained by using splitting in single pass are given in Table 4.3. The results obtained by using splitting in two passes are given in Table 4.4. The results obtained by using splitting in single pass for ER-related classifiers and two pass for others are given in Table 4.5. The results obtained by using best splitting for each individual classifier are given are given in Table 4.6.

Table 4.3: Results for splitting in one pass

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**84.54**	4.14	4.66	6.64
Mito.	13.27	**63.27**	14.45	8.99
Cyt.	5.55	15.98	**46.19**	32.26
Nuc.	3.32	13.58	22.51	**60.56**

Table 4.4: Results for splitting in two passes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**75.27**	12.43	5.27	7.02
Mito.	14.30	**68.98**	8.35	8.35
Cyt.	5.09	14.40	**45.55**	34.95
Nuc.	3.87	11.22	16.00	**68.90**

Table 4.5: Results for splitting in one pass for ER classifiers, splitting in two passes for others

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**83.39**	6.09	4.47	6.03
Mito.	12.67	**70.14**	9.71	7.46
Cyt.	5.68	13.70	**46.49**	34.11
Nuc.	3.25	11.46	17.51	**67.76**

Table 4.6: Results for using best splitting method

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**82.77**	4.75	5.07	7.39
Mito.	14.36	**66.71**	12.01	6.90
Cyt.	5.93	10.87	**49.80**	33.38
Nuc.	2.95	9.71	17.63	**69.70**

4.4.2 Second Fold

The results obtained by using splitting in single pass are given in Table 4.7. The results obtained by using splitting in two passes are given in Table 4.8. The results obtained by using splitting in single pass for ER-related classifiers and two pass for others are given in Table 4.9. The results obtained by using best splitting for each individual classifier are given are given in Table 4.10.

Table 4.7: Results for splitting in one pass

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**78.42**	11.18	5.55	4.83
Mito.	6.95	**75.10**	13.63	4.31
Cyt.	5.07	16.50	**55.03**	23.39
Nuc.	2.05	11.23	28.79	**57.92**

Table 4.8: Results for splitting in two passes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**75.91**	13.08	6.15	4.84
Mito.	9.45	**68.28**	16.68	5.56
Cyt.	4.84	15.49	**53.67**	25.98
Nuc.	3.11	13.24	28.64	**55.00**

Table 4.9: Results for splitting in one pass for ER classifiers, splitting in two passes for others

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**78.32**	10.50	6.41	4.75
Mito.	7.05	**70.94**	15.65	6.34
Cyt.	5.30	15.90	**52.53**	26.26
Nuc.	2.10	12.51	28.11	**57.26**

Table 4.10: Results for using best splitting method

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**77.36**	10.89	7.37	4.37
Mito.	6.78	**70.69**	18.99	3.52
Cyt.	5.24	13.36	**60.48**	20.90
Nuc.	2.47	9.41	31.63	**56.46**

4.4.3 Third Fold

The results obtained by using splitting in single pass are given in Table 4.11. The results obtained by using splitting in two passes are given in Table 4.12. The results obtained by using splitting in single pass for ER-related classifiers and two pass for others are given in Table 4.13. The results obtained by using best splitting for each individual classifier are given are given in Table 4.14.

Table 4.11: Results for splitting in one pass

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**71.68**	15.18	7.59	5.54
Mito.	5.38	**71.03**	14.70	8.87
Cyt.	2.42	13.23	**54.10**	30.23
Nuc.	2.22	11.62	29.44	**56.70**

Table 4.12: Results for splitting in two passes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**76.24**	11.87	6.49	5.38
Mito.	11.00	**69.87**	13.45	5.65
Cyt.	5.08	15.57	**54.01**	25.32
Nuc.	4.40	12.07	32.42	**51.10**

Table 4.13: Results for splitting in one pass for ER classifiers, splitting in two passes for others

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**73.90**	13.68	7.12	5.28
Mito.	5.75	**72.93**	13.99	7.30
Cyt.	2.39	14.86	**55.24**	27.49
Nuc.	2.27	12.06	32.48	**53.17**

Table 4.14: Results for using best splitting method

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**76.75**	11.65	6.49	5.09
Mito.	5.22	**69.95**	17.41	7.40
Cyt.	2.65	13.90	**56.27**	27.17
Nuc.	1.92	9.40	26.94	**61.71**

4.4.4 Fourth Fold

The results obtained by using splitting in single pass are given in Table 4.15. The results obtained by using splitting in two passes are given in Table 4.16. The results obtained by using splitting in single pass for ER-related classifiers and two pass for others are given in Table 4.17. The results obtained by using best splitting for each individual classifier are given are given in Table 4.18.

Table 4.15: Results for splitting in one pass

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**74.04**	11.45	11.24	3.26
Mito.	6.95	**71.06**	19.14	2.83
Cyt.	3.06	19.09	**67.94**	9.89
Nuc.	2.87	16.56	40.94	**39.61**

Table 4.16: Results for splitting in two passes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**79.38**	10.34	7.52	2.74
Mito.	3.19	**74.83**	17.97	3.99
Cyt.	3.44	26.06	**56.29**	14.19
Nuc.	3.22	16.38	38.11	**42.27**

Table 4.17: Results for splitting in one pass for ER classifiers, splitting in two passes for others

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**72.43**	14.93	9.36	3.25
Mito.	6.90	**70.84**	17.74	4.50
Cyt.	3.21	26.88	**55.37**	14.52
Nuc.	2.91	18.08	36.73	**42.26**

Table 4.18: Results for using best splitting method

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**80.61**	9.13	6.13	4.11
Mito.	4.31	**75.39**	15.96	4.31
Cyt.	4.35	16.79	**60.83**	18.01
Nuc.	3.02	13.60	34.81	**48.55**

The mean and standard deviation values of the four fold results for each different initialization technique applied can be found at Appendix D.

4.5 Comparison of the Results with Other Systems

We compared the results obtained for 3 classes with TargetP [14]. TargetP is also a subcellular localization prediction tool which is based on neural networks. It is designed for 3-class classification where classes are ER, Mitochondria and Other. The results for both systems are given in Table 4.19.

Table 4.19: Comparison of TargetP and our system

Actual		Predicted		
		ER	Mito.	Other
ER	Our Sys.	**85.36**	1.58	13.06
	TargetP	**85.21**	2.06	12.73
Mito.	Our Sys.	9.61	**66.39**	24.00
	TargetP	3.64	**77.84**	18.52
Other	Our Sys.	7.10	8.69	**84.21**
	TargetP	1.88	8.86	**89.27**

We compared the results obtained for 4 classes with P2SL [1]. P2SL is also a subcellular localization prediction tool which is similar to our system. It is designed for the same 4 classes in our system. The results for both systems are given in Table 4.20.

Table 4.20: Comparison of P2SL and our system

Actual		Predicted			
		ER	Mito.	Cyt.	Nuc.
ER	Our Sys.	**83.88**	2.46	3.47	10.17
	P2SL	**83.97**	3.99	8.21	3.81
Mito.	Our Sys.	7.95	**70.42**	7.95	13.66
	P2SL	6.59	**75.45**	16.02	1.93
Cyt.	Our Sys.	6.72	7.15	**33.18**	52.92
	P2SL	3.81	3.56	**79.68**	12.93
Nuc.	Our Sys.	3.62	5.69	11.35	**79.31**
	P2SL	3.51	3.10	30.10	**63.27**

4.6 Discussion on the Results

Considering the results presented in this chapter, we see that:

- In a 4-class environment, the system can recognize the ER targeted proteins with a high rate but the rate drops for proteins targeting other localizations.
- In a 3-class environment, the system can recognize the ER targeted proteins

and proteins targeting other than ER and Mitochondria with a high rate but the rate drops for proteins targeting Mitochondria.

- Considering the results of 3-class and 4-class version, it can be deduced that the system confuses whether a protein is targeting Cytoplasm or Nucleus which causes the lower recognition rate of Cytoplasmic and Nuclear proteins.

- The two pass clustering method works up to 9 times faster than the normal clustering method without losing significant precision. Using more proteins than normal spectral clustering in training, however, did not increase its success rate.

- Considering these facts, it can be deduced that the success of distinguishing between Cytoplasmic and Nuclear proteins does not depend much on the number of the input data used for training.

- The results obtained from our system are comparable to the results of TargetP. Our system has a very similar recognition rate for ER and Other classes compared to TargetP. But the recognition rate of our system is lower in Mitochondrial proteins.

- The results obtained from our system are comparable to the results of P2SL. Our system has a very similar recognition for ER targeted and Nuclear proteins compared to P2SL. For Cytoplasmic proteins, our system has a recognition rate of half of P2SL. The recognition rate of our system is higher in Mitochondrial proteins. It must be noted that, however, P2SL predicts the second possible localization, if any, and for most of the Nuclear proteins P2SL predicted to target Cytoplasm, it predicted the second localization as Nucleus.

- We believe that introducing the insertion and deletion operations in string edit distance would increase the success in case of Cytoplasmic and Nuclear proteins. This idea is supported by the fact that the P2SL system has a higher success rate using the same amino acid index and decomposition parameters [1].

Chapter 5

Conclusion

5.1 Summary and Discussion

In this work, a method to predict subcellular localization of a protein based on its sequence was described. Subsequences of a protein sequence were employed as basic unit of information. During the prediction process, spectral clustering was the major computational tool to determine which subsequences were similar to each other. Hence, similar subsequences were grouped by spectral clustering. Feature vector was formed using the relations between the subsequences of a protein sequence and these groups. Feature vector was then fed to a classifier.

The use of spectral clustering is one of the most important aspects of this work and this is one of the first studies to use spectral clustering in subcellular localization. Spectral clustering is a robust clustering algorithm and its application is quite suitable to the prediction of subcellular localization. It has two basic parameters both of which can be determined from the clustering results. More importantly spectral clustering does not depend on the data domain. Its success is independent of the data domain itself and it does not impose constraints on the data unlike many other clustering algorithms. Most of the clustering algorithms require an update or merge ability for the data. In our particular case this corresponds to creating a new subsequence using two or more subsequences. Since subsequences, or amino acids, do

not correspond to numerical values, this is a constraint that cannot be fulfilled easily or completely. Spectral clustering does not require any change on the data during clustering so such a requirement does not exist. Spectral clustering only requires a distance metric to be defined on the data domain which actually is a necessity for any clustering algorithm. Spectral clustering's path retrieval ability also fits to our problem. Spectral clustering can retrieve clusters where each element is not very close but has path of close elements connecting them. With the correct distance measurement techniques, spectral clustering creates meaningful biological clusters.

The major problem in using spectral clustering is its time and memory complexity. Because of this fact, small increases in input size causes large increases in completion time and memory requirements at the training phase. Long training times in spectral clustering makes parameter search difficult. In order to decrease the time complexity of the spectral clustering algorithm, we proposed a new method which is based on splitting the data into subsets and clustering them individually. This method clusters the data in two passes which gives it its name, Two Pass Spectral Clustering. Although the asymptotic complexity of this method is the same as the original method, we saw that it runs dramatically faster for inputs of suitable sizes for our particular problem. We also observed that we did not lose precision while gaining speed. This method gave similar success rates as the normal method.

Two Pass Spectral Clustering requires its input to be splitted into subsets which introduces a new problem. This splitting is for initialization purposes and requires to be fast as well as stable. We proposed a method for solving this problem which is based on hierarchical clustering with single linkage but is simpler in merging strategies. Due to the bounded size requirement, this method creates subsets where their sizes are bounded which gives this method its name, Bounded Hierarchical Clustering.

To sum up the results, the following can be said:

- The system can recognize the ER targeted proteins and protein targeting other than ER and Mitochondria as a single class with a high rate.

- Recognition rate drops for proteins targeting Mitochondria.
- The system also confuses whether a protein is targeting Cytoplasm or Nucleus.
- The sytem has comparable results with well-known subcellular localization prediction tools like TargetP and P2SL.
- Introduction of insertion and deletion operations to string edit distance used may increase the recognition rate of Cytoplasmic and Nuclear proteins. We believe that the confusion among Cytoplasmic and Nuclear proteins is due to the lack of these operations.
- The two pass spectral clustering method proposed in this work is faster than the normal spectral clustering. This speed is gained without losing significant precision.

5.2 Future Work

We will now briefly discuss on the possible future research topics to improve the system.

An important work to be done is the introduction of gap cost in string edit distance which allow insertion and deletion operations. This is necessary to compute the distance between long subsequences, such as subsequences of 15 aminoacids, accurately in biological sense.

The majority voting algorithm deciding the actual class using the results from binary classifiers can be improved. The system can be improved so that the results of classifiers will be weighed according their accuracy in training phase. The more accurate a classifier is, the higher its weight will be. The weighting system can be designed in such a fashion that the unselected label for a classifier will also receive some votes. This way the failure probability of the classifiers will be taken into account.

The main cause of the high time complexity of the problem is due to the fact that eigenvalue decomposition has a high time complexity. Complexity of eigenvalue decomposition can be decreased if the matrix subject to decomposition is sparse. The distance metric can be altered so that distance values drop to 0 fast and the affinity matrix used in spectral clustering becomes sparse. In this case this case techniques for eigenvalue decomposition in sparse matrices can be used which are faster than the original eigenvalue decomposition.

In feature vector generation, for each subsequence the closest cluster to it is computed and the subsequence is decided to be a member of that cluster. This can be changed to a probabilistic approach where a subsequence can be a member of more than one cluster with membership weight describing the probability of the subsequence being in that cluster which sums up to 1.

Appendix A

K-Means Clustering

Introduction

K-Means algorithm is a popular algorithm for data clustering. It can be considered as a member of the EM algorithms family. K-Means takes the Euclidean Sum of Squares as the objective function and attempts to minimize it. It achieves this goal by minimizing the intra-cluster distance while as a side-effect maximizing the inter-cluster distances. The error function is

$$Error = \sum_i \sqrt{\sum_d (\hat{x}_d^i - \hat{w}_d^j)^2}$$

where $\hat{x}^i$'s are data points and $\hat{w}^j$'s are the means of the clusters the respective point assigned to. By minimizing this function, K-Means clustering assigns points to closer clusters and thus minimizes the sum of the square distances.

Method

The traditional K-Means clustering procedure is given in Algorithm 5.

Algorithm 5 K-Means Clustering

```
1: choose k initial points as cluster means
2: repeat
3:    for all points in the data set do
4:       compute the distance to the mean of each cluster
5:       set its new cluster as the nearest cluster
6:    end for
7:    recompute cluster means
8: until all cluster means are the same as the previous iteration {a stable k-cluster
   partition has been generated}
```

Initialization and Convergence

It is proven that K-Means actually converges in case of using Euclidean Distance as the distance metric[7]. But it is not guaranteed that the method will converge to the global minimum. Usually a certain run converges to a local minima and gets stuck there. Actually this depends mainly on the choice of initial clusters means. A simple solution to this problem is to make a number of runs with different initial conditions and select the best result, that is the result with minimal error among all.

The selection of initial cluster means for a simple run is another problem (and is currently beyond the scope this research). Some obvious solutions to this problem can be

- Selecting means completely in a random fashion
- Selecting means from the data in a random fashion
- Selecting means from the data according to a appropriately selected distribution (where this method can be considered as a generalized version of the previous method).
- Selecting means using the results of previous runs of K-Means.

Appendix B

Sample Amino Acid Similarity Index

This is the *Isomorphicity of Replacements* matrix used in examples and in system test through-out this work.

[100, −2, −8, 3, 4, 0, 12, −3, −10, −17, −6, 2, −2, −8, −2, 10, −3, −12, −18, −12]

[−2, 100, 11, 2, −12, 26, 11, −6, 0, −25, −24, 24, 0, −11, −4, −10, −5, −7, 0, −18]

[−8, 11, 100, 32, −7, 12, 13, 10, 8, −32, −35, 17, −12, −23, 8, 1, 12, −6, −13, −30]

[3, 2, 32, 100, −20, 8, 34, 8, 10, −39, −40, 10, −16, −31, 1, 7, 11, −20, −22, −32]

[4, −12, −7, −20, 100, −8, −18, −1, 3, 12, 15, −14, 2, 11, −3, −6, −4, 8, 5, 8]

[0, 26, 12, 8, −8, 100, 37, −10, −4, −20, −26, 26, −12, −22, 0, −7, −10, −3, −23, −12]

[12, 11, 13, 34, −18, 37, 100, −5, −8, −36, −40, 24, −17, −40, −7, −10, −12, −10, −28, −20]

[−3, −6, 10, 8, −1, −10, −5, 100, −8, −21, −16, −11, −10, −12, 4, 9, −13, −13, −18, −14]

[−10, 0, 8, 10, 3, −4, −8, −8, 100, 3, −9, −4, −8, −10, −2, −3, −7, −6, 12, −10]

[−17, −25, −32, −39, 12, −20, −36, −21, 3, 100, 46, −17, 18, 32, −18, −17, −5, 12, 29, 46]

[−6, −24, −35, −40, 15, −26, −40, −16, −9, 46, 100, −33, 24, 36, −16, −11, −11, 8, 16, 26]

[2, 24, 17, 10, −14, 26, 24, −11, −4, −17, −33, 100, −19, −26, −2, −10, −8, −10, −21, −14]

[−2, 0, −12, −16, 2, −12, −17, −10, −8, 18, 24, −19, 100, 20, −10, −10, −8, 12, 8, 10]

[−8, −11, −23, −31, 11, −22, −40, −12, −10, 32, 36, −26, 20, 100, −2, 0, 4, 8, 24, 16]

[−2, −4, 8, 1, −3, 0, −7, 4, −2, −18, −16, −2, −10, −2, 100, 15, 2, −9, −9, −18]

[10, −10, 1, 7, −6, −7, −10, 9, −3, −17, −11, −10, −10, 0, 15, 100, 24, −18, −16, −16]

[−3, −5, 12, 11, −4, −10, −12, −13, −7, −5, −11, −8, −8, 4, 2, 24, 100, −8, −2, −4]

[−12, −7, −6, −20, 8, −3, −10, −13, −6, 12, 8, −10, 12, 8, −9, −18, −8, 100, 6, 14]

[−18, 0, −13, −22, 5, −23, −28, −18, 12, 29, 16, −21, 8, 24, −9, −16, −2, 6, 100, 17]

[−12, −18, −30, −32, 8, −12, −20, −14, −10, 46, 26, −14, 10, 16, −18, −16, −4, 14, 17, 100]

Appendix C

Sample Converted Amino Acid Similarity Index

If A is the original similarity index, max is the maximal number in A, min is the minimal number in A and $range$ is $max - min$ then the converted index A' which reflects the distances between amino acids is as follows:

$$A'_{ij} = range - (A_{ij} - min)$$

The converted version of the similarity index in Appendix B is as follows:

[0, 102, 108, 97, 96, 100, 88, 103, 110, 117, 106, 98, 102, 108, 102, 90, 103, 112, 118, 112]

[102, 0, 89, 98, 112, 74, 89, 106, 100, 125, 124, 76, 100, 111, 104, 110, 105, 107, 100, 118]

[108, 89, 0, 68, 107, 88, 87, 90, 92, 132, 135, 83, 112, 123, 92, 99, 88, 106, 113, 130]

[97, 98, 68, 0, 120, 92, 66, 92, 90, 139, 140, 90, 116, 131, 99, 93, 89, 120, 122, 132]

[96, 112, 107, 120, 0, 108, 118, 101, 97, 88, 85, 114, 98, 89, 103, 106, 104, 92, 95, 92]

[100, 74, 88, 92, 108, 0, 63, 110, 104, 120, 126, 74, 112, 122, 100, 107, 110, 103, 123, 112]

[88, 89, 87, 66, 118, 63, 0, 105, 108, 136, 140, 76, 117, 140, 107, 110, 112, 110, 128, 120]

[103, 106, 90, 92, 101, 110, 105, 0, 108, 121, 116, 111, 110, 112, 96, 91, 113, 113, 118, 114]

[110, 100, 92, 90, 97, 104, 108, 108, 0, 97, 109, 104, 108, 110, 102, 103, 107, 106, 88, 110]

[117, 125, 132, 139, 88, 120, 136, 121, 97, 0, 54, 117, 82, 68, 118, 117, 105, 88, 71, 54]

[106, 124, 135, 140, 85, 126, 140, 116, 109, 54, 0, 133, 76, 64, 116, 111, 111, 92, 84, 74]

[98, 76, 83, 90, 114, 74, 76, 111, 104, 117, 133, 0, 119, 126, 102, 110, 108, 110, 121, 114]

[102, 100, 112, 116, 98, 112, 117, 110, 108, 82, 76, 119, 0, 80, 110, 110, 108, 88, 92, 90]

[108, 111, 123, 131, 89, 122, 140, 112, 110, 68, 64, 126, 80, 0, 102, 100, 96, 92, 76, 84]

[102, 104, 92, 99, 103, 100, 107, 96, 102, 118, 116, 102, 110, 102, 0, 85, 98, 109, 109, 118]

[90, 110, 99, 93, 106, 107, 110, 91, 103, 117, 111, 110, 110, 100, 85, 0, 76, 118, 116, 116]

[103, 105, 88, 89, 104, 110, 112, 113, 107, 105, 111, 108, 108, 96, 98, 76, 0, 108, 102, 104]

[112, 107, 106, 120, 92, 103, 110, 113, 106, 88, 92, 110, 88, 92, 109, 118, 108, 0, 94, 86]

[118, 100, 113, 122, 95, 123, 128, 118, 88, 71, 84, 121, 92, 76, 109, 116, 102, 94, 0, 83]

[112, 118, 130, 132, 92, 112, 120, 114, 110, 54, 74, 114, 90, 84, 118, 116, 104, 86, 83, 0]

Appendix D

Mean and Standard Deviations for Four Folds

This section contains the mean and standard deviation values of the four fold results for each different initialization technique applied.

D.1 Values for Splitting in Single Pass

The mean values for splitting in single pass are given in Table D.1. The standard deviation values for splitting in single pass are given in Table D.2.

Table D.1: Mean values for splitting in one pass

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**77.17**	10.49	7.26	5.07
Mito.	8.14	**70.12**	15.48	6.25
Cyt.	4.03	16.2	**55.82**	23.94
Nuc.	2.62	13.25	30.42	**53.7**

Table D.2: Standard deviation values for splitting in one pass

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**5.65**	4.61	2.92	1.42
Mito.	3.5	**4.95**	2.48	3.15
Cyt.	1.52	2.4	**9**	10.11
Nuc.	0.59	2.44	7.68	**9.53**

D.2 Values for Splitting in Two Passes

The mean values for splitting in two passes are given in Table D.3. The standard deviation values for splitting in two passes are given in Table D.4.

Table D.3: Mean values for splitting in two passes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**76.7**	11.93	6.36	5
Mito.	9.49	**70.49**	14.11	5.89
Cyt.	4.61	17.88	**52.38**	25.11
Nuc.	3.65	13.23	28.79	**54.32**

Table D.4: Standard deviation values for splitting in two passes

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**1.83**	1.17	0.93	1.77
Mito.	4.66	**2.97**	4.29	1.81
Cyt.	0.79	5.48	**4.7**	8.5
Nuc.	0.6	2.26	9.37	**11.08**

D.3 Values for Splitting in One Pass for ER Classifiers, Splitting in Two Passes for Others

The mean values for splitting in single pass for ER-related classifiers and two pass for others are given in Table D.5. The standard deviation values for splitting in single pass for ER-related classifiers and two pass for others are given in Table D.6.

Table D.5: Mean values for splitting in one pass for ER classifiers, splitting in two passes for others

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**77.01**	11.3	6.84	4.83
Mito.	8.09	**71.21**	14.27	6.4
Cyt.	4.15	17.84	**52.41**	25.6
Nuc.	2.63	13.53	28.71	**55.11**

Table D.6: Standard deviation values for splitting in one pass for ER classifiers, splitting in two passes for others

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**4.94**	3.94	2.02	1.18
Mito.	3.11	**1.2**	3.41	1.36
Cyt.	1.6	6.1	**4.16**	8.15
Nuc.	0.54	3.07	8.25	**10.54**

D.4 Values for Using Best Splitting Method

The mean values for using best splitting for each individual classifier are given are given in Table D.7. The standard deviation values for using best splitting for each individual classifier are given are given in Table D.8.

Table D.7: Mean values for using best splitting method

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**79.37**	9.11	6.27	5.24
Mito.	7.67	**70.69**	16.09	5.53
Cyt.	4.54	13.73	**56.85**	24.87
Nuc.	2.59	10.53	27.75	**59.11**

Table D.8: Standard deviation values for using best splitting method

Actual	Predicted			
	ER	Mito.	Cyt.	Nuc.
ER	**2.83**	3.09	0.95	1.49
Mito.	4.58	**3.58**	2.99	1.91
Cyt.	1.42	2.43	**5.13**	6.84
Nuc.	0.51	2.05	7.48	**8.9**

References

[1] V. Atalay and R. Çetin Atalay. Implicit motif distribution based hybrid computational kernel for sequence classification. *Bioinformatics*, 21(8):1429–1436, 2005.

[2] T.K. Attwood and M.E. Beck. Printsa protein motif fingerprint database. *Protein Engineering*, 7(7):841–848, 1994.

[3] F. R. Bach and M. I. Jordan. Learning spectral clustering. Technical Report UCB/CSD-03-1249, Computer Science Division (EECS), University of California, Berkeley, California 94720, June 2003.

[4] T. L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

[5] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S. R. Eddy, S. Griffiths-Jones, K. L. Howe, M. Marshall, and E. L. L. Sonnhammer. The pfam protein families database. *Nucleic Acids Research*, 30(1):276–280, 2002.

[6] K. Blekas, D. I. Fotiadis, and A. Likas. Greedy mixture learning for multiple motif discovery in biological sequences. *Bioinformatics*, 19(5):607–617, 2003.

[7] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithms. *Advances in Neural Information Processing Systems*, 7:585–592, 1995.

[8] P. Bucher and A. Bairoch. A generalized profile syntax for biomolecular sequences motifs and its function in automatic sequence interpretation. In *Proceed-*

ings of the 2nd International Conference on Intelligent Systems for Molecular Biology, pages 53–61.

[9] F. Chang. *Spectral Graph Theory*, chapter 1, pages xii–207. AMS Publications, 2. edition, 1997.

[10] G. Cormode. *Sequence Distance Embeddings*. PhD thesis, University of Warwick, January 2003.

[11] J. H. Dierendonck, I. Bahar, A. R. Atilgan, R. L. Jernigan, and B. Erman. Understanding the recognition of protein structural classes by amino acid composition. *Proteins: Structure, Function, and Genetics*, 29(2):172–185, 1997.

[12] R. Van Driessche and D. Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing*, 21(1):29–48, January 1995.

[13] O. Emanuelsson. Predicting protein subcellular localisation from amino acid sequence information. *Briefings in Bioinformatics*, 3(4):361–376, 2002.

[14] O. Emanuelsson, H. Nielsen, S. Brunak, and G. Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *Journal of Molecular Biology*, 300(4):1005–1016, 2000.

[15] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23:298–305, 1973.

[16] Igor Fischer and Jan Poland. New methods for spectral clustering. Technical Report IDSIA-12-04, IDSIA, Dalle Molle Institute for Artificial Intelligence Galleria 2, 6928 Manno, Switzerland, June 2004.

[17] J. L. Gardy, C. Spencer, K. Wang, M. Ester, G. E. Tusndy, I. Simon, S. Hua, K. deFays, C. Lambert, K. Nakai, and F. S. L. Brinkman. Psort-b, improving protein subcellular localization prediction for gram-negative bacteria. *Nucleic Acids Research*, 31(13):36133617, 2003.

[18] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[19] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19(23):6565–6572, 1991.

[20] C. W. Hsu and C. J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[21] L. Hunter, editor. *Artificial Intelligence and Molecular Biology*, chapter 1, pages 1–46. MIT Press, March 1993.

[22] R. Kannan, S. Vempala, and A. Vetta. On clusterings - god, bad and spectral. *FOCS*, pages 367–377, 2000.

[23] Z. Lu, D. Szafron, R. Greiner, P. Lu, D. S. Wishart, B. Poulin, J. Anvik, C. Macdonell, and R. Eisner. Predicting subcellular localization of proteins using machine-learned classifiers. *Bioinformatics*, 20(4):547–556, 2004.

[24] M. Meilă and J. Shi. A random walks view of spectral segmentation. 2001.

[25] B. Mohar. The laplacian spectrum of graphs. In *Graph theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.

[26] B. Mohar. Some applications of laplace eigenvalues of graphs. In G. Hahn and G. Sabidussi, editors, *Graph Symmetry: Algebraic Methods and Applications*, volume NATO ASI Ser. C 497, pages 225–275. Kluwer Academic, 1997.

[27] R. Mott, J. Schultz, P. Bork, and C. P. Ponting. Predicting protein cellular localization using a domain projection method. *Genome Research*, 12(8):1168–1174, 2002.

[28] R. Nair and B. Ros. Loc3d: annotate sub-cellular localization for protein structures. *Nucleic Acids Research*, 31(13):3337–3340, 2003.

[29] K. Nakai. Protein sorting signals and prediction of subcellular localization. *Adv Protein Chem.*, 54:277–344, 2000.

[30] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14, 2001.

[31] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[32] D. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science*, pages 96–105, Los Alamitos, CA, 1996. IEEE Comput. Soc. Press.

[33] G. V. Stewart and Ji Guang Sun. *Matrix Perturbation Theory.* Academic Press, 1990.

[34] M. Stoer and F. Wagner. A simple min-cut algorithm. *ACM*, 44(4):585591, 1997.

[35] D. Verma and M. Meilă. Comparison of spectral clustering methods. Technical report, Department of Statistics, University of Washington, Seattle, WA 98195, 2005.

[36] U. von Luxburg. A tutorial on spectral clustering. Technical Report 149, Max Planck Institute, 2006.

[37] D. Wagner and F. Wagner. Between min-cut and graph bisection. In *Proceedings of The 18th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 744–750, 1993.

[38] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 975–982, 1999.