2015

Predicting Protein Contact Map By Bagging Decision Trees

Chuqiao Ren
Bucknell University, cr025@bucknell.edu

PREDICTING PROTEIN CONTACT MAP BY BAGGING DECISION TREES

by

Chuqiao Ren

A Thesis

Presented to the Faculty of

Bucknell University

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science with Honors in Computer Science

April 24, 2015

Approved: Brian R. King

Brian R. King
Thesis Advisor

[illegible]

Luiz Felipe Perrone
Thesis Advisor

[illegible]

Stephen Guattery
Chair, Department of Computer Science

Acknowledgments

This thesis would not be possible without the support of many people. I would like to devote this page to thank these individuals for their encouragement and belief in me through my research and life at Bucknell:

- First and foremost, Dr. Brian R. King, for his patience, guidance and support in my research and life. Without your knowledge and your brilliant ideas, I will not be able to finish this project and thesis. You are the one who introduced me to the world of computer science and bioinformatics, and you have guided me not only on research, but also on life and career. I have enjoyed all the times we have spent together; you are my greatest mentor and friend. Thank you very much for everything.
- Professor Pamela Gorkin, for her guidance and support on my mathematics. It is impossible for me to pursue a mathematics major without your encouragement. Your courses are among the best classes I have taken through my college career, and I hope you can keep challenging us. I cannot thank you enough.
- Professor L. Felipe Perrone and Professor Sharon Garthwaite, for helping me understand the thesis defense process and taking the time out of your busy schedules to be my faculty readers.
- Professor Steve Guattery, for being my third faculty reader and helping me with graduate application.
- Tinge Gu, for giving me enormous support over the last four years. I cannot achieve all these goals without your accompany.
- My parents, for all your encouragement and guidance through my life. I love you.
- Jinbo Wang, Tingyi Jin, and Yipeng Huang, for being my best friends and roommates and supporting me through my research, study and life.

Table of Contents

List of Tables

List of Figures

Abstract

Proteins' function and structure are intrinsically related. In order to understand proteins' functionality, it is essential for medical and biological researchers to determine proteins' three-dimensional structure. The traditional method using NMR spectroscopy or X-ray crystallography are inefficient compared to computational methods. Fortunately, substantial progress has been made in the prediction of protein structure in bioinformatics. Despite these achievements, the computational complexity of protein folding remains a challenge. Instead, many methods aim to predict a protein contact map from protein sequence using machine learning algorithms. In this thesis, we introduce a novel ensemble method for protein contact map prediction based on bagging multiple decision trees. A random sampling method is used to address the large class imbalance in contact maps. To generalize the feature space, we further clustered the amino acid alphabet from twenty to ten. A software is also developed to view protein contact map at certain threshold and separation. The parameters used in decision trees are determined experimentally, and the overall results for the first L, $L/2$ and $L/5$ predictions for protein of length L are evaluated.

Chapter 1

Introduction and Background

It is essential for medical and biological researchers to understand proteins' three-dimensional (3D) structure, for proteins' function and structure are intrinsically related. However, over the past 30 years, determination of proteins' 3D structure are hindered by the inefficiency of traditional experimental methods such as NMR spectroscopy or X-ray crystallography. At the same time, the influx of a huge number of completed genomes resulting from advanced DNA sequencing techniques further deepen the protein sequence-structure gap [1]. By 2015 there were 46,714,516 protein sequences predicted by translating the existing genes from DNA sequence, but only 104,000 of these proteins are structurally determined based on experimental methods and recorded in Protein Database (PDB) by 2014 [2]. Some even claim that it is impossible to determine protein structures solely by experiments [3]. For these reasons researchers must explore new methods with higher efficiency that can determine the structure of proteins.

This chapter will introduce bioinformatics, an emerging interdisciplinary study in computer science and biology, and machine learning, a branch of artificial intelligence that utilizes statistics to recognize patterns in data. Furthermore, this chapter will give an insight on protein, including its composition, structure and representation.

1.1 Protein

Proteins are the most fundamental building blocks of organisms. Proteins are large and complex molecules that play many critical roles in the body. DNA polymers encode all the information to make proteins. DNA is transcribed to mRNA, and then mRNA is translated by ribosome to amino acid sequences. Protein's 3D structure is intrinsically related to its function. Thus it is essential for medical and biological researcher to understand proteins' 3D structure. A protein must correctly fold into its 3-dimensional entity in order to properly function. Incorrect protein folding can lead to serious diseases such as cancer. Accurate fold determination requires X-ray Crystallography or NMR Spectroscopy; however, they are slow and expensive. Currently, researchers are working toward computational methods to predict a protein fold from sequence, but this still remains an unsolved problem, and represents one of the grand challenges in biological and biomedical research today.

1.1.1 Amino Acids

There are 20 amino acids in nature. An amino acid is composed of an amine (-NH_2) and a carboxylic (-COOH) functional group, and a unique side-chain. An amino acid possesses multiple physical-chemical properties, and many amino acids share similar properties. Hydrophilicity and hydrophobicity, for example, are two physical-chemical properties that are widely used to categorize amino acids. Polarity is another example of a physical-chemical property; amino acids are neutral (no charge), positively charged or negatively charged. Because all of the physical-chemical properties of amino acids influence the protein folding process, they have significant impact on the proteins' overall 3D structure.

1.1.2 Protein Structure

Protein has four levels of structure. Its primary structure is its sequence of amino acids. Two amino acids are connected by a peptide bond. The secondary structure of a protein has many forms, with the two most common being alpha helix and beta sheet. They are formed by the hydrogen bond connecting two carboxy groups from two amino acids. Protein's 3D structure is the tertiary structure of a protein. If multiple proteins act together and form a group, this cluster of protein is named the protein's quaternary structure.

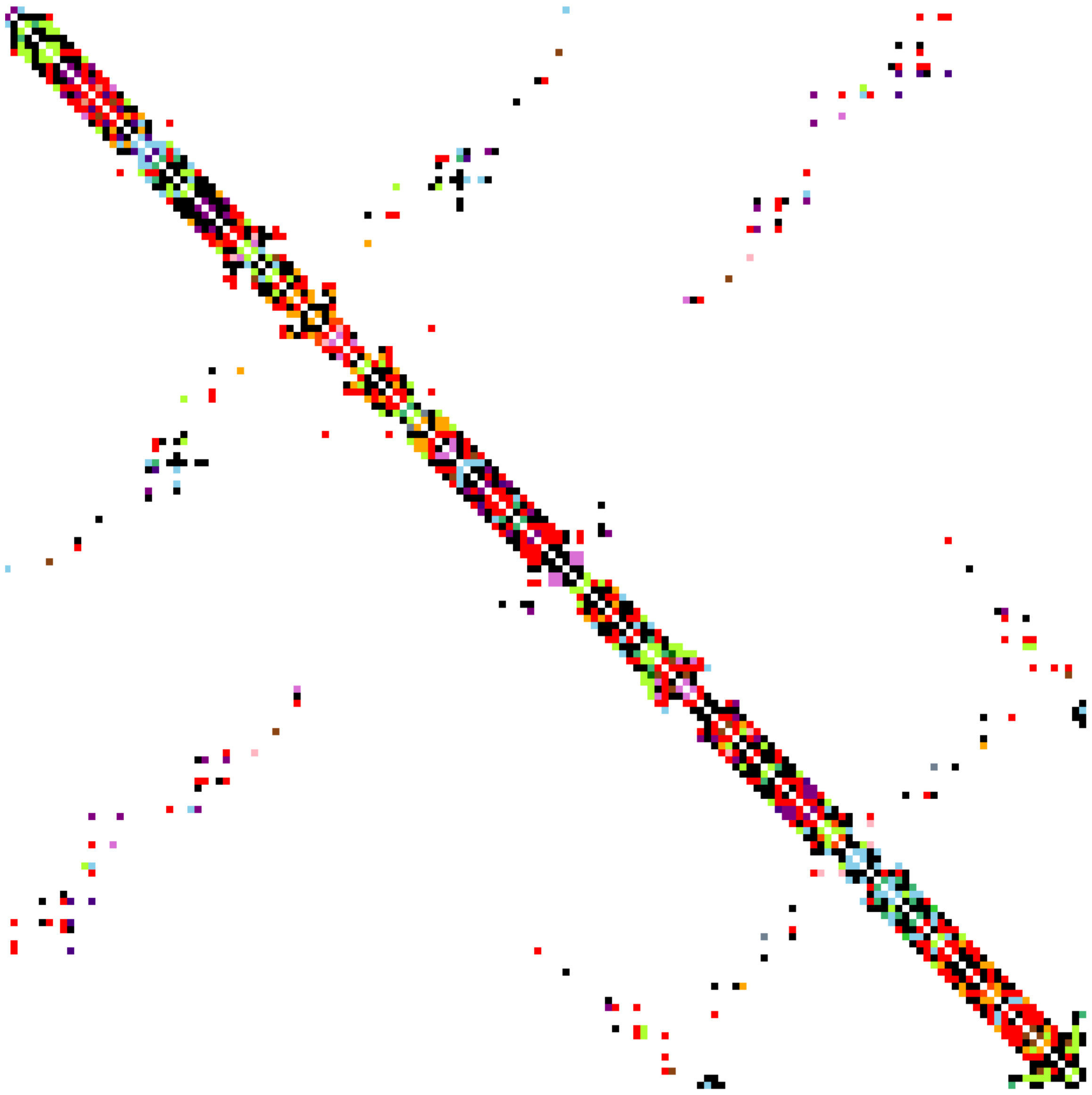


Figure 1.1: A protein contact map of sequence A in protein 1MBO at threshold of 8.

1.1.3 Protein Contact Map

The protein contact map (PCM) of a protein of length L is a $L \times L$ binary matrix. It is a simplified version of distance matrix. A distance matrix captures the distance between alpha carbons [1] of all amino acids within a given protein sequence in three-dimensional space. The alpha carbon (C_α) of an amino acid is the first carbon that attaches to the functional group. Suppose $\mathbf{v} := (x_1, y_1, z_1)$ is the three-dimensional position of C_α in a_i, and $\mathbf{w} := (x_2, y_2, z_2)$ is that of a_j; then the ij^{th} entry in the distance matrix is the Euclidean distance between each two C_α, and is calculated as follows:

$$e_{ij} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}.$$

Two amino acids a_i and a_j are *in contact* if the distance between their alpha carbons is less than threshold t (normally $t = 8\text{Å}$). Therefore, the ij^{th} entry of PCM is 1 if $e_{ij} \leq t$, meaning a_i and a_j are in contact; otherwise, the ij^{th} entry of PCM is 0. Because $e_{ij} = e_{ji}$, PCM is symmetric along the diagonal [4].

1.2 Bioinformatics

Bioinformatics is an emerging field of study that applies computer science methods to solve biological problems. This unexpected union between computer science and biology is because life itself can be viewed as an information technology – organisms are determined by genes, which one could view as digital information [5]. There are three major aims of bioinformatics. First, to organize and categorize the enormous biological data allow researchers to quickly access existing information and to easily

[1] In this project we are interested in distance matrix that captures the distances between alpha carbons, but this is not necessary. Besides alpha carbons in amino acids, distance matrix can also be calculated from the distances between beta carbons, the second carbon connecting to the functional group in an amino acid.

submit new entries. For example, the Protein Database (PDB) is a publicly available database containing amino acid composition and three-dimensional (3D) structure of a large amount of proteins. The second aim is to develop tools and methods that can help researchers to analyze and understand biological data. For example, PSI-Blast [6] is a tool developed by National Center for Biotechnology Information (NCBI) to align and compare a given protein sequence to a set of protein sequences. The third aim of bioinformatics is to use these tools and methods to analyze a huge amount of biological data. While biologists and chemists examine systems and organisms individually in detail, researchers in bioinformatics are able to extract patterns across many organisms simulateously [5].

Protein structure prediction is one of the most classical problems in this field. Protein is the most essential material in almost all organisms, and its functionality and 3-D structure are closely related. To understand organisms, researchers must study protein's function, and hence determination of protein structure is significant and urgent. Bioinformatic methods use machine learning algorithms to predict protein structure. Compared to experimental methods using X-ray crystallography and NMR spectroscopy, these computational methods are more efficient yet less accurate. Because of the computational difficulties in 3D structure comparison, researchers reduce dimensionality by projecting 3D structure onto 2D space. This projection is reversible; 2D maps of distances between residues allows the reconstruction of the full 3D structure. To further simplify the computation, most of the data assume a binary representation of the 2D distance matrix, a.k.a the protein contact map.

There are two types of methods for computational prediction of protein contact maps. The template-based modeling (TBM) methods use the existing protein structure as a template to predict protein contact maps. Nevertheless, these methods require the existence of similar structures in the PDB, which is usually not the case in most *ab initio* predictions. Threading is the other approach to protein structure prediction, which uses techniques including sequence profile-profile alignments (PPAs), structural profile alignments, hidden Markov models (HMMs), machine learning, and others [7]. Though not a single method in the threading approach can outperform traditional and TBM methods, this approach is more general and inexpensive, and more importantly, it has a large potential to be improved.

1.3 Machine Learning

Learning in general is a process "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience," and "modification of a behavioral tendency by experience" [8]. Machine learning focus on building computer programs that can acquire new knowledge or to improve already possessed knowledge based on input information. Machine learning is a core area in artificial intelligence. Since learning is necessary in any kind of problem solving, machine learning can be used in various contexts such as decision making, classification, pattern recognition and task execution. Machine learning methods are especially valuable in extracting important relationships and correlations hidden in large datasets. A typical machine learning process includes two stages, training and testing. The training data contains information to build models, and the test data is used to evaluate performance.

There are two common problems that machine learning methods focus on: *classification* and *clustering*. *Classification* is a form of data analysis that induces a model (or *classifier*) to classify data. Each datum in the training data has a discrete and unordered class label. The classifier aim to predict the class label of each datum in the testing set. On the other hand, the data in the *clustering* problems have no class labels. Given a large number of dataset and many attributes describing this dataset, some interesting patterns can be extracted by grouping data with high similarities into subsets (or *clusters*) [9]. Besides extracting information from data, clustering can also be used to detect outliers.

The following sections introduce some important methods in machine learning, including the decision tree algorithm, bagging and hierarchical clustering. The decision tree algorithm is the primary method used in this research; it is a fundamental yet powerful classification technique in machine learning. Bagging is a machine learning ensemble algorithm aiming to boost performance in statistical learning. Hierarchical clustering is another way to classify data using clustering.

1.3.1 Decision Tree

Decision Tree Induction

Decision trees are popular methods in classification. The decision tree consists of:

- root and internal nodes, splitting on attributes that lead to the least impurity
- leaf or terminal nodes, indicating classes

An *attribute selection measure* is a way to select the *splitting criterion* that partitions given data at root and internal nodes [9]. There are many variations of decision trees; in this project we used Quinlan's C4.5 decision tree [10]. Suppose D is a training set of tuples with m distinct classes, denoted C_i for $i = 1, \ldots, m$. Let node N represent the tuples of D. The expected *information* needed to classify a tuple in D is:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

where $p_i = \frac{|C_i|}{|D|}$ is the probability that a tuple belongs to class C_i [9]. Let $A = \{a_1, a_2, \ldots, a_v\}$ denote the set of v attributes as observed in training data. The information we need to classify for attribute A is:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j),$$

where D_j contains tuples in D that have outcome a_j of A [9]. Then *information gain* is defined as:

$$Gain(A) = Info(D) - Info_A(D),$$

and it provides us how much information would be gained by splitting D on attribute A [9]. To determine the *splitting criterion*, we need to find attribute A such that it has the maximum $Gain(A)$. After a decision tree is built, we are able to classify the test data. During the testing phase, every input datum from test set will start at root node, go through internal nodes and reach the leaf node that contains a certain class label. This test datum will then be collected in this class.

Pruning the Tree

There are two approaches of pruning: *prepruning* and *postpruning*. In the *prepruning* approach, the tree construction is halted based on some specified criterion [9]. A common approach constrains the number of instances required at a leaf. Let *minNode* be the minimum number of instances per leaf. The tree will stop splitting if the number of instances per leaf is less than *minNode*. Therefore, a larger *minNode* values indicates a more general decision tree.

In the *postpruning* approach, the tree is allowed to fully grow but some branches of the tree will be removed after the construction of tree is done [9]. *Cost complexity* is a common measure in the *postpruning* approach. The *cost complexity* is a function of the number of leaves and the *error rate* of the tree, where *error rate* is the percentage of misclassified tuples [9]. Starting from the bottom, each subtree will be evaluated based on a *cost complexity* function, and will be pruned (replaced by a leaf node) if the *cost complexity* is greater than the expected value [9]. Hence, a smaller expected *cost complexity* value will output a more general tree.

1.3.2 Artificial Neural Networks

An artificial neural network (ANN) is a method in classification inspired by the human brain. *Neurons*, the nerve cells in human brains, connect to each other via strands of fiber called *axons*. The network of neurons enable us to learn. Analogous to the human brain, an ANN is composed of nodes and links. The simplest model of an ANN is called the *perceptron*, in which the output node translates information from input nodes with weight. In research, an ANN always has multiple layers of perceptrons: the output from the input layer will be directed to the hidden layer(s), and the results from hidden layer(s) will be summarized by the output layer. The goal of an ANN model is to find a vector of weights that minimize the total sum of squared errors in the network [11]. After adding hidden layer(s), the minimization becomes more complicated. A technique know as *back-propagation* has been developed to solve this problem. This method uses errors for neurons at layer $k+1$ to estimate errors at layer k. One of the drawbacks of ANN is that at the beginning of the training phase, one must specify initial values of weights, which are always hard to estimate and initialize correctly.

1.3.3 Support Vector Machine

Support vector machines (SVM) are methods in classification rooted in statistical learning theory. An SVM produces nonlinear boundary by constructing a linear decision boundary in a transformed version of the feature space. Suppose we have a data set of N training data, and each datum is represented as $(\mathbf{x_i}, \mathbf{y_i})$, where $\mathbf{x_i}$ is a vector of attributes (or input features) and $\{x_{i1}, x_{i2}, \dots x_{im}\}^T$ is the attribute set of the i^{th} data, and y_i is the class label for data i. Now the decision boundary of a linear classifier is defined as $\mathbf{w} \cdot \mathbf{x} + \mathbf{b}$, where $\mathbf{w}$ is a vector of weights and b is the bias [11]. The function $\Phi(\mathbf{x})$ projects the vector of attributes to a high-dimensional space, in which a linear decision boundary with the form $\mathbf{w} \cdot \Phi(\mathbf{x}) + \mathbf{b} = \mathbf{0}$ can be found separating the transformed space. The learning task of a nonlinear SVM is to optimize

$$\min_{\mathbf{w}} \frac{||\mathbf{w}||^{\mathbf{2}}}{2}$$

subject to $y_i(\mathbf{w} \cdot \Phi(\mathbf{x_i}) + \mathbf{b}) \geq \mathbf{1}$ for $i \in \{1, 2, \dots, N\}$ [11].

Due to the high cost of computing the dot product on transformed data tuples, a similarity function, K, is designed to substitute the dot product in the transformed space. This is commonly known as the *kernel function*. Suppose there are two input vectors, $\mathbf{u}$ and $\mathbf{v}$. Then we have [11]:

$$K(\mathbf{u}, \mathbf{v}) \equiv \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + \mathbf{1})^{\mathbf{2}}.$$

The kernel function is directly applied to the original input data, and it can replace $\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$ in the training algorithm [9]. There are various forms of kernel functions; *radial basis kernel* (RBF) is one of them. When using the RBF, $K(\mathbf{u}, \mathbf{v})$ is defined as

$$K(\mathbf{u}, \mathbf{v}) = e^{-\gamma||\mathbf{u}-\mathbf{v}||^2},$$

where $\gamma = \frac{1}{\sigma^2}$ [4]. From the training set, SVM learns a classification function $f(x)$ as

$$f(x) = \sum_{i=1}^{N} w_i K(\mathbf{x}, x_i) + b,$$

where w_i is the weight assigned to x_i from $\mathbf{w}$ [11, 4]. For test instance $\mathbf{t}$, SVM is able to classify it by the output from

$$f(\mathbf{t}) = sign(\mathbf{w} \cdot \Phi(\mathbf{z}) + b).$$

1.3.4 Bagging

Bagging, or bootstrap aggregating, is an ensemble algorithm that is able to stabilize the classifier and hence improve the overall performance of machine learning methods. Suppose a dataset D of d tuples contains m training examples. A subset D_i of d tuples will be sampled from D with replacement at the i^{th} iteration, where $i = 1, 2, \ldots, k$ [9]. This process is called the *bootstrap replicate* [12]. Because bagging samples data with replacement, some of the data from D may not be included in any subsets while some of the data may be included multiple times. Independent from d, k and m, the resulting *bootstrap replicate* contains 63.2% of the original data set in theory, with several data examples appearing in multiple times [12]. A classifier M_i is learnt for each subset D_i. To classify a testing tuple t, each classifier will output a class label, and the overall class label for t will be determined by majority vote. This technique can significant improve the accuracy of any classifier.

1.3.5 Hierarchical Clustering

Hierarchical clustering is an important category of clustering methods. Typically, there are two types of hierarchical clustering methods: *agglomerative* and *divisive*. Agglomerative hierarchical clustering starts with individual clusters and merges the closest pair of clusters, while divisive hierarchical clustering methods split a single, all-inclusive cluster into clusters that contains only individual points. The graphical representation of hierarchical clustering is called a *dendrogram*, which visually represents the relationship between individual clusters. Agglomerative hierarchical clustering is used in this project. The distance or proximity between two clusters is used to determine whether two clusters should be joined or not. Smaller distance implies two clusters are closer.

There are four common methods to determine distance between two clusters: single linkage, complete linkage, group average and centroid [11]. Single linkage calculates proximity based on the distance between the closest points in different clusters. Proximity in complete linkage, on the other hand, is determined by the distance between the furtherest points in different clusters. The group average method uses the average distance in between all points within two clusters as the proximity measurement, and the centroid method uses distance from centroids to centroids as proximity. The complete linkage is used to construct agglomerative hierarchical clusters in this project.

Chapter 2

Related Work

As computational resources have progressed, so have the methods for dealing with protein structures and their fold. Researchers continue to develop methods that can successfully predict proteins' 3D structure from its amino acid sequence. Unfortunately, the complexity of the protein folding problem is still beyond what modern computers can handle. In fact, not a single prediction method can achieve equivalent results for predicting a protein fold compared to experimental methods [4]. Compared to amino acid sequences, a protein contact map, which is a two-dimensional (2D) distance matrix between residues, contains more valuable structural information. For computational simplicity, the distances between residues are further simplified as binary contacts [1]. This simplified 2D matrix, commonly known as a protein contact map (PCM), can be used to recover the 3D structure of a protein [13], and thus the problem of how to accurately predict unknown protein's PCM emerges. There are many existing sequence-based approaches to this problem, and most of them use machine learning algorithms, such as decision tree [14], artificial neural networks [1, 4] and support vector machines [15].

2.1 Decision Tree

2.1.1 DTP

DTP is a decision tree-based solution to predict protein contact map. The training data set of DTP includes distance and sequence separation between amino acids along with the frequency of amino acids in the subsequence. The training set is fed to Quinlan's C4.5 decision tree [10], and a 10-fold cross validation is used to evaluates the performance. The overall dataset contains protein chains with the lowest homology possible and to control overfitting. The performance was reported over groups of proteins with different length ranges. The average accuracy of all proteins for this method was 0.34, and this algorithm performed better if the protein length was bigger than 300 [14].

DTP uses a distance matrix as a basis to train the predictor, while other methods mostly used binary residual contact matrix as their training set. The distance matrix encloses more detailed informations of amino acids' location, but is more complex than the binary residual contact map, making training the decision tree computationally prohibitive. DTP induced decision trees using Quinlan's C4.5 decision tree algorithm with the default settings for all possible pairs of contacts.

2.2 Artificial Neural Networks

2.2.1 PROFcon

PROFcon used artificial neural networks to construct a protein contact map model. The training set contains 633 protein chains directly extracted from PDB with known structures. Training data is fed to a standard feed-forward neural network with back-propagation [1]. Because of the symmetry property of protein contact map, predictions of ij and ji should be the same. This was enforced by PROFcon in a way that the output value of ij and ji was the average predicted probability of ij and ji [1]. The results from PROFcon are based on annotations from the Structural Class of Proteins (SCOP) [16].

SCOP categorizes all proteins to four types according to their secondary structures: all-alpha, all-beta, alpha+beta and alpha/beta. All-alpha protein contains only alpha helices while all-beta protein solely beta sheets. Proteins in alpha+beta category have alpha helices and antiparallel beta-sheets while those in alpha/beta have alpha helices and parallel beta-sheets. PROFcon performed the best on proteins in alpha+beta category, with an accuracy of 0.360 and recall of 0.10 [1].

Separation of amino acids on protein sequence is another factor fed into PROFcon. Nearby amino acids on protein sequence are more likely to form contacts, but these contacts are usually not in any secondary structures. Hence it is better to filter out these contacts at the data-preprocessing stage. PROFcon performed the best at separation 6; that is, for amino acid at position (i, j), all contact and non-contact entries within window spans $\{(i-3)(j-3), (i-3)(j+3), (i+3)(j-3), (i+3)(j+3)\}$ will be removed from training data. Protein length also has an impact on the performance; the longer the protein is, the lower the accuracy and recall are. At the optimal separation 6, PROFcon performs the best at protein length 201-300, with accuracy of 0.339 and recall of 0.086. The overall accuracy is 0.324 and recall is 0.098 at the optimal separation 6. All values are calculated for the first $L/2$ predictions, where L is the length of each protein.

2.2.2 CNNcon

CNNcon is another method using Neural Networks to construct model and predict protein contact map. Because the contact density decreases as protein sequence length increases, it is harder to predict contact maps on medium- and long-range proteins [4]. In order to maintain stability and consistency over all protein length, CNNcon implements a cascade-network of six sub-networks for different protein length range [4]. All six sub-networks are all feed-forward neural networks with back-propagation, and each neural network has three layers composed of 1747 input nodes, 5 hidden nodes and 1 output node [4]. A different sub-network is trained and tested using different data sets based on protein sequence length, and the outputs are collected together for evaluation purpose.

Because of the linear relationship in between protein sequence length and number of contacts in a protein, CNNcon assesses the performance based on the average on the results from fractions of the total true contacts of the whole test data set [4]. The average accuracy of sub-networks is 0.3401 with a recall of 0.3544, and the overall accuracy, after combining results from all sub-networks, is 0.5786 with a recall of 0.3428 [4].

2.3 Support Vector Machine

2.3.1 SVMcon

SVMcon uses support vector machines (SVM) to predict medium- and long-range contacts. It uses a RBF kernel, with $\gamma = 0.025$. The input features are from five categories for each pair of residues at position (i, j) on protein contact map [15], including information from protein sequence itself, multiple sequence alignment, amino acid's physicochemical properties, residual separations and protein length. The training dataset contains 485 sequences and the test set has 48 sequences [15]. Since SVMcon is designed for medium- and long-range contacts, only medium- and long-range residual pairs with sequence separation greater or equal to 6 are extracted from dataset [15]. The threshold of protein contact map is 8Å. SVMcon uses protein's secondary structure information from SCOP structure class to classify proteins. Due to variance of performance between individual proteins, the authors report a full list of accuracy and recall for each protein in the test set. The per-protein accuracy ranges from 0.042 to 0.630 and recall from 0.019 to 0.446 at separation ≥ 6 (the optimal separation). SVMcon performs better on proteins containing beta-sheets (beta, alpha+beta and alpha/beta) than those having alpha-helices [15].

2.4 Summary

In this chapter, we introduced a number of existing methods aiming to predict protein contact maps using different machine learning algorithms. **DTP** is a Decision Tree-based predictor, similar to the one we proposed in the following chapter. However, DTP is not among the best predictors. While other methods, including our proposed method, feed protein contact maps to the model, DTP uses distance matrix as a basis for training [14]. Distance matrix contains more detailed information about amino acid residues than PCM, but the computational costs for training and testing are relatively high. For the DTP model, types of contacts are classified using the following criteria:

$$contact \leq 8\text{Å} < quasi - contact \leq 12\text{Å} < non - contact,$$

but accuracy and recall are calculated based on number of contacts and non-contacts, leaving quasi-contact undefined [14]. Furthermore, DTP uses default settings of Quinlan's C4.5 tree, which might not be the optimal settings within this context.

CNNcon and **PROFcon** are two methods using neural networks to predict protein contact map [1, 4]. They used distance between beta carbons, C_{beta}, to determine whether two amino acids are in contact or not with standard threshold, 8Å. Though

CNNcon performs better than PROFcon, CNNcon uses a slightly different evaluation method.

Chapter 3

Methods

In this chapter, we will explain how datasets are prepared and a model is induced for prediction of protein contact maps. To compare our result with that of the existing methods, we use a dataset that has been widely used in the literature. Due to the nature of protein contacts, the data is highly imbalanced toward non-contacts; therefore, we employ a resampling technique to solve this problem. A hierarchical clustering method is used to cluster amino acids, reducing the alphabet of amino acids from 20 to 10, based on similarity among amino acids' physicochemical properties. To have a better understanding of protein contact map, we develop software to visualize protein contact maps with the ability to show amino acid clusters. The model in this chapter is first proposed by Ren and King, a novel meta-method that combines multiple decision trees using bagging [17]. We evaluate performance using identical techniques in literature, based on the number of predicted contacts that is proportional to the protein sequence length.

3.1 Datasets

We assessed datasets from Griep and Hobohm's PDBselect list, a list of proteins with known 3D structures [18]. To compare our result with that of existing methods, we extracted the same set of protein IDs using the exact method present by CNNcon [4]. The dataset was first generated using the default *nsigma* value (3.0). We filtered all protein sequences that were not determined by X-ray. We further selected protein chains with 25% threshold and resolution $\leq$ 1.5Å [18]. The dataset, after all the filtering, has 4350 proteins. Because our algorithm is designed to predict short-range proteins, we focused on 109 proteins that have length range from 50 to 80.

3.2 Visualizing Protein Contact Map

We developed software to visualize protein contact maps. Users can easily manipulate distance and separation values after entering the PDB identifier of a protein. The amino acid contacts are assigned to different colors other than black if two amino acids share the same chemical physical properties by default. Users can also choose to show protein contact map after clustering, and amino acid contacts are assigned to colors other than black if two amino acids fall into the same cluster.

3.3 Preparing Data

In order to reduce the feature space, we used hierarchical clustering to cluster amino acids over their physicochemical properties. This reduced the number of amino acids clusters from 20 (one amino acid per cluster) to 10. We constructed protein contact map (PCM) from distance matrices using a threshold of 8Å and separation of 6 for all 109 protein sequences. To capture the local features of PCM, we used a 5×5 sliding window to generate .arff file, a standard file format used in Weka.

3.3.1 Amino Acid Clustering

There are 20 amino acids in nature, but many amino acids share similar physicochemical properties. In order to reduce the attribute space of the data, we used hierarchical clustering with complete linkage to cluster amino acids over 12 major properties that may have substantial influence on how a protein folds, including hydrophobicity, polarity and size [19].

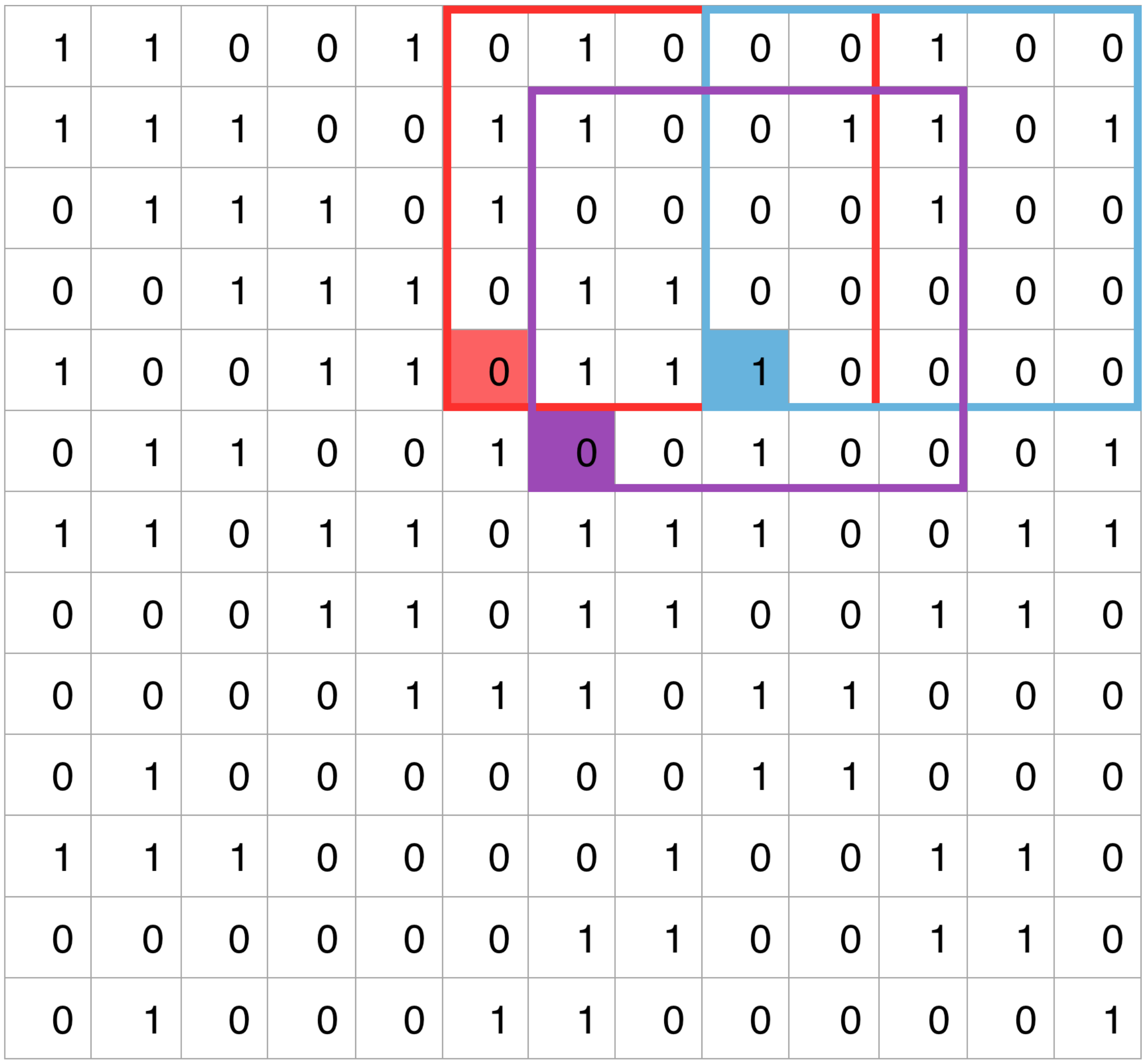
3.3.2 Construct Protein Contact Map

Distance matrices for all 109 proteins are calculated from files downloaded directly from Protein Data Bank (PDB), an online database containing information of proteins with known 3D structures. Protein Contact Maps (PCM) are produced from distance matrices with threshold of 8Å and separation of 6. Amino acids used in both distance matrix and PCM are based on our clusters identified from the original set of 20 amino acids as described in **Section 1.1.3**.

3.3.3 Sliding Window

After we generate PCM for all proteins, the data is divided into small matrices by sliding a 5×5 window across the contact map. In other words, for the ij^{th} entry the of PCM where $i > 2$ and $j > 2$, the window will span the set $\{(i-2)(j-2), (i-2)(j+2), (i+2)(j-2), (i+2)(j+2)\}$. To capture the amino acid properties, the set amino acid clusters that spans row $\{(i-2)(j-2), (i+2)(j-2)\}$ and that spans column $\{(i-2)(j-2), (i-2)(j+2)\}$ will be recorded in the output file. Suppose C_{mn} denotes the mn^{th} entry in the small matrix. The cell $C_{40} = (i-2, j+2)$ (lower left cell) is set as the target class where $i, j \in \mathbb{Z}^+$ and $i, j \geq 2$; that is, the contact (0 or 1) of C_{40} will be recorded in the output file. The output file is in the .arff format, a standard format used by Weka, as shown in **Figure 3.2**.

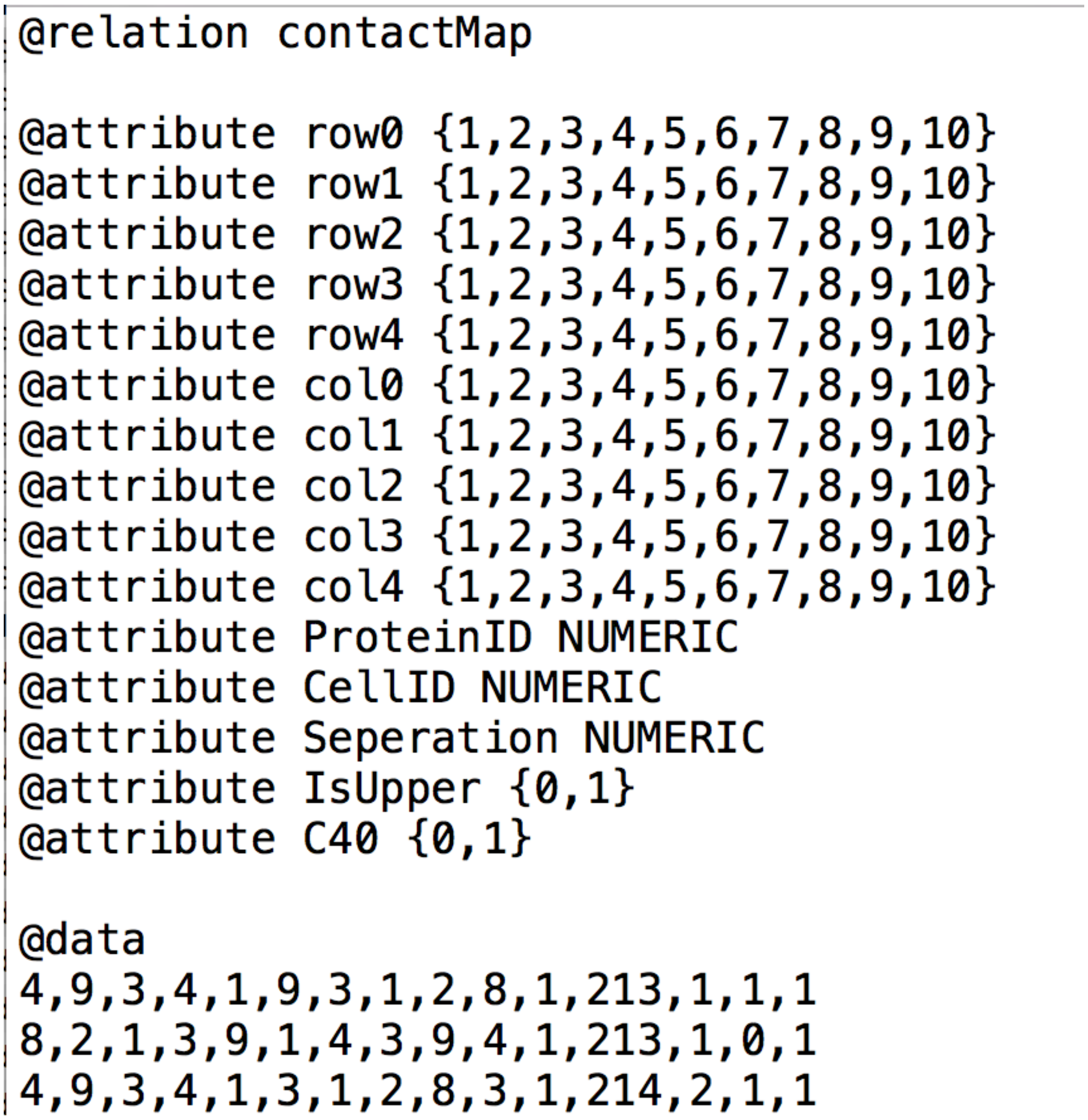
Because a PCM is symmetric, the predictions should be the same across the diagonal line. We denote set U spans $\{(i,i), (i, i+L), (i+L, i), (i+L)(i+L)\}$ be the upper part of a given PCM, where $0 < i < L$ is an integer and L is the protein length. The sliding window will only be valid if $C_{40} \in U$. The attributes of a sliding window, s, at the upper part of a given PCM traverse from $i-2$ to $i+2$ and from $j-2$ to $j+2$. To include information from lower part of a PCM, for the same sliding window s, we replicate the entry from upper part of PCM by changing attributes of the same class label. The attributes of a lower part will traverse from $j+2$ to $j-2$ and from



1	1	0	0	1	0	1	0	0	0	1	0	0
1	1	1	0	0	1	1	0	0	1	1	0	1
0	1	1	1	0	1	0	0	0	0	1	0	0
0	0	1	1	1	0	1	1	0	0	0	0	0
1	0	0	1	1	0	1	1	1	0	0	0	0
0	1	1	0	0	1	0	0	1	0	0	0	1
1	1	0	1	1	0	1	1	1	0	0	1	1
0	0	0	1	1	0	1	1	0	0	1	1	0
0	0	0	0	1	1	1	0	1	1	0	0	0
0	1	0	0	0	0	0	0	1	1	0	0	0
1	1	1	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	1	0	0	0	0	0	1

Figure 3.1: This is an example of sliding window. The sliding window will start from red window, move to blue window and then go to purple window. C_{40} cell is the class label. The C_{40} cell of each window has the same color as the window boarder.

$i + 2$ to $i - 2$ for sliding window s. **Figure 3.1** shows how to extract data from a PCM using sliding window.



```
@relation contactMap

@attribute row0 {1,2,3,4,5,6,7,8,9,10}
@attribute row1 {1,2,3,4,5,6,7,8,9,10}
@attribute row2 {1,2,3,4,5,6,7,8,9,10}
@attribute row3 {1,2,3,4,5,6,7,8,9,10}
@attribute row4 {1,2,3,4,5,6,7,8,9,10}
@attribute col0 {1,2,3,4,5,6,7,8,9,10}
@attribute col1 {1,2,3,4,5,6,7,8,9,10}
@attribute col2 {1,2,3,4,5,6,7,8,9,10}
@attribute col3 {1,2,3,4,5,6,7,8,9,10}
@attribute col4 {1,2,3,4,5,6,7,8,9,10}
@attribute ProteinID NUMERIC
@attribute CellID NUMERIC
@attribute Seperation NUMERIC
@attribute IsUpper {0,1}
@attribute C40 {0,1}

@data
4,9,3,4,1,9,3,1,2,8,1,213,1,1,1
8,2,1,3,9,1,4,3,9,4,1,213,1,0,1
4,9,3,4,1,3,1,2,8,3,1,214,2,1,1
```

Figure 3.2: This is an example of the .arff output file. The first line define the class, and the lines following tell Weka the formats of attributes. The lines after data indicator is entries collected by sliding window from PCM.

3.4 Data Preprocessing

Due to the nature of our dataset, there are many more non-contacts than contacts, resulting in highly unbalanced classes. We used a resampling technique to address this issue. Our resampling technique will generate a random subsample of our dataset *without replacement* to increase the representation of contacts in each sample. Given a dataset of contacts $\mathbb{D}$, a sample of $\mathbb{D}$, denoted $\mathbb{S}_k$, is generated by selecting all contacts, and a random sample of non-contacts without replacement. Sample $\mathbb{S}_k$ is formed such that it is 20% of the size of $\mathbb{D}$.

3.5 Model

In this section, we propose a novel ensemble method of decision tree. We used the concept of bagging to boost the performance of decision tree algorithm. We used *leave-one-out* cross validation to evaluate the performance. The overall dataset D is divided such that each subset, D_i, contains only one protein, where $i = 1, 2, \ldots, 109$ in this project. In the i^{th} iteration, we designate subset D_i as the testing set and the rest subsets as the training set. For each iteration, we construct 100 decision trees, and collect prediction over these trees with probabilistic threshold of contacts ranging from 0 to 1. Because of the linear relationship between the number of contacts and protein length, we follow the standard way to assess model performance – the number of contacts is based on the first $L/2$ or $L/5$ predictions.

3.5.1 Decision Tree

The $J48$ decision tree method in Weka was used as the base decision tree implantation, which is basically the same as Quinlan's C4.5 tree [10]. The confidence factor in pruning is 0.5. To ensure the generality of the tree, the minimum number of instances per leaf is set to 25. We further constrain the number of splits from each node to be 2 (*binary splits*).

3.5.2 Bagging Decision Tree

The sparseness of amino acid contacts results in a dataset that is highly imbalanced. The resampling method improves the class balance. To further improve performance, we designed an enhancement of bagging to maximize the predictive power of the minority class. A single tree T_k is induced from S_k, the subset generated by resampling. This process is repeated 100 times, resulting in 100 different decision trees by varying seed values. A prediction is generated for each entry from each T_k. The average probability of each residue pair being in contact is aggregated over all 100 trees, and this probability will be compared to 100 different thresholds. If the average probability is bigger than threshold probability, the predicted output will be 1; otherwise, the output will be 0. All the predictions from each fold are combined and model performance is assessed.

3.6 Evaluating the Performance of a Classifier

3.6.1 Cross Validation

Once the classifier model has been successfully constructed, it can be applied to a test dataset. Normally, records in the test dataset have correct labels, to which the output from the classifier model will be compared. Cross-validation is a commonly used technique to estimate the generalization error of a given classifier. In cross-validation, the whole dataset D is equally partitioned into k subsets. The training-testing cycle will run k times. Suppose $i \in \{1, 2, \ldots, k\}$. At the i^{th} run, subset D_i will be the designated test set, and the rest of the subsets will serve as training sets. The total errors will be found by summing up the errors for all k runs [11]. Now suppose the dataset D contains N data points. The extreme case of cross-validation sets $k = N$, which is usually called the *leave-one-out* approach. In this thesis, k is set to N.

3.6.2 Performance Measurements

Many measures are used to assess the performance of classifiers. *True positive* (TP), *false positive* (FP), *true negative* (TN) and *false negative* (FN) are four basic representations indicating the discrepancy between predicted outcome and the true value. TP is the number of correctly classified tuples of positive class, and FN is the correctly classified tuples of negative class. FP and TN is the falsely classified tuples in positive class of negative class, respectively. *Accuracy* (or *specificity*) is defined as follows:

$$Acc = \frac{TP}{TP + FP},$$

where $TP + FP$ is the number of predicted contacts. Though accuracy evaluates the percentage of true predictions over all positive predicted outcomes, it is not suffice to use it alone. One should contrast it to the *recall* (or *sensitivity*), defined as:

$$Rec = \frac{TP}{TP + FN},$$

where $TP + FN$ is the number of actual contacts in the data. One way to assess the contradiction between accuracy and recall is to use F_1 score, defined as follows:

$$F_1 = \frac{2TP}{2TP + FP + FN}.$$

We evaluate performance on a number of predicted contacts that is proportional to protein length L. The results for the first $L/2$ and $L/5$ predictions for each protein are reported along with the full length result.

3.6.3 ROC Curve and AUC

Receiver operating characteristic curve (ROC curve) is a useful tool to compare the result of different classifiers by showing the tradeoffs between the *true positive rate* (TPR) and *false positive rate* (FPR) [9]. The TPR is the proportion of positive tuples that are correctly classified and can be calculated as follows:

$$TPR = \frac{TP}{TP + FN};$$

similarly, FPR is the proportion of negative tuples that are misclassified as positive, and can be calculated as

$$FPR = \frac{FP}{FP + TN}.$$

For a random model, $FPR = TPR$ and the ROC curve is a straight line with slope 1. If the ROC curve of a classifier is below the random ROC curve, then this classifier performs worse than random. The *area under curve* (AUC) is a relatively precise measure of the accuracy of the classifier [9]. A greater AUC indicates a better classifier.

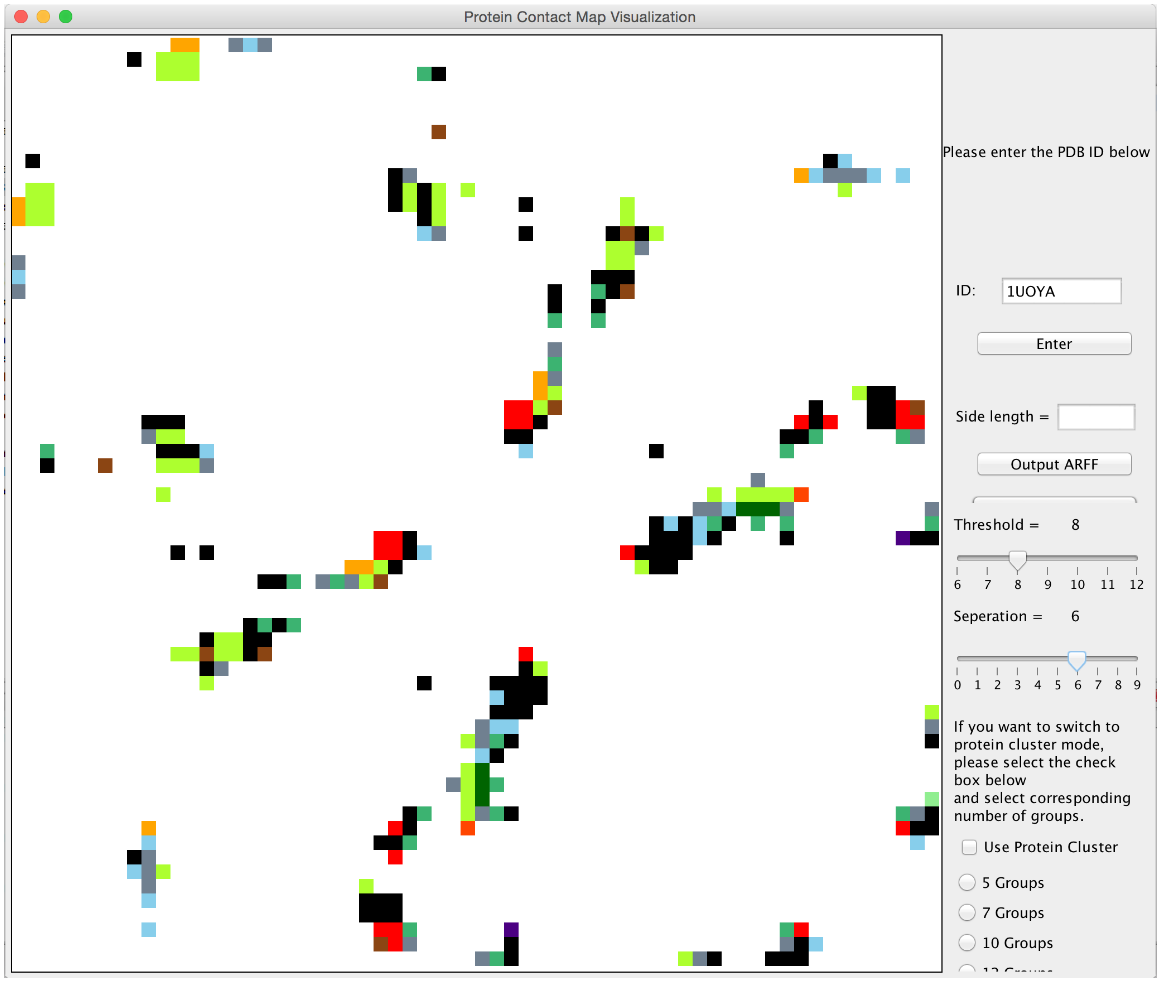
Chapter 4

Results

To have a better understanding of protein contact maps (PCM), we present PCMs using the software we developed in Java. Our software allows users to view a PCM by entering protein ID on PDB. Proteins can be viewed under different threshold, separation and amino acid clustering values. We will also show the clustering result of amino acids. To evaluate the performance of our model, we used Leave-One-Out validation. ROC curves are generated over all predictions for each protein separately, and these ROC curves are combined to output an average ROC curve. Using these ROC curves and basic statistics output from our model, we are able to determine the parameters of decision trees that will give us the most optimal result. We experimentally determined that our model performed the best with sample size = 20% and minNode = 20. For a protein of length L, the top $L/2$ and $L/5$ predictions are assessed for accuracy and recall [15].

4.1 Visualization of PCM

We developed software that enables users to view a PCM in a more directed and interactive way. **Figure 4.1** is an example of a contact map of a bubble protein from *penicillium breviconpactum dierckx exudate* (PDBID: 1UOY) drawn by our software. The panel on the left shows a PCM as a matrix with the protein sequence aligned along the x-axis and y-axis. If a pair of amino acids, a_{ij} for example, shares the same physicochemical property, the entry at position (i, j) will be colored a specific color designated to this physicochemical property. If not, the entry will be black. To view a PCM, the user must enter a protein's Protein Database (PDB) ID number (i.e. $A, B, \ldots, Z$) on the control panel to the right of the viewing panel. The default threshold is 6 with separation 0 and 20 clusters. The user can change these settings on the control panel. The user can output a training/testing file in .arff file format using the sliding window technique as described in **Section 3.3.3** by entering the preferred window size, and clicking on the *Output-ARFF* button. The *Output-Frequency* button will output an *Excel* sheet containing the frequency of each amino acid in the protein sequence shown in the viewing panel. **Figure 4.2** shows the PCM of protein $1MBO$ with 7 clusters. In the cluster mode, the color code used to present different physiochemical properties in the viewing panel is different from the normal mode.



Please enter the PDB ID below

ID: 1UOYA

Side length =

Threshold = 8

Seperation = 6

If you want to switch to protein cluster mode, please select the check box below and select corresponding number of groups.

Use Protein Cluster

5 Groups

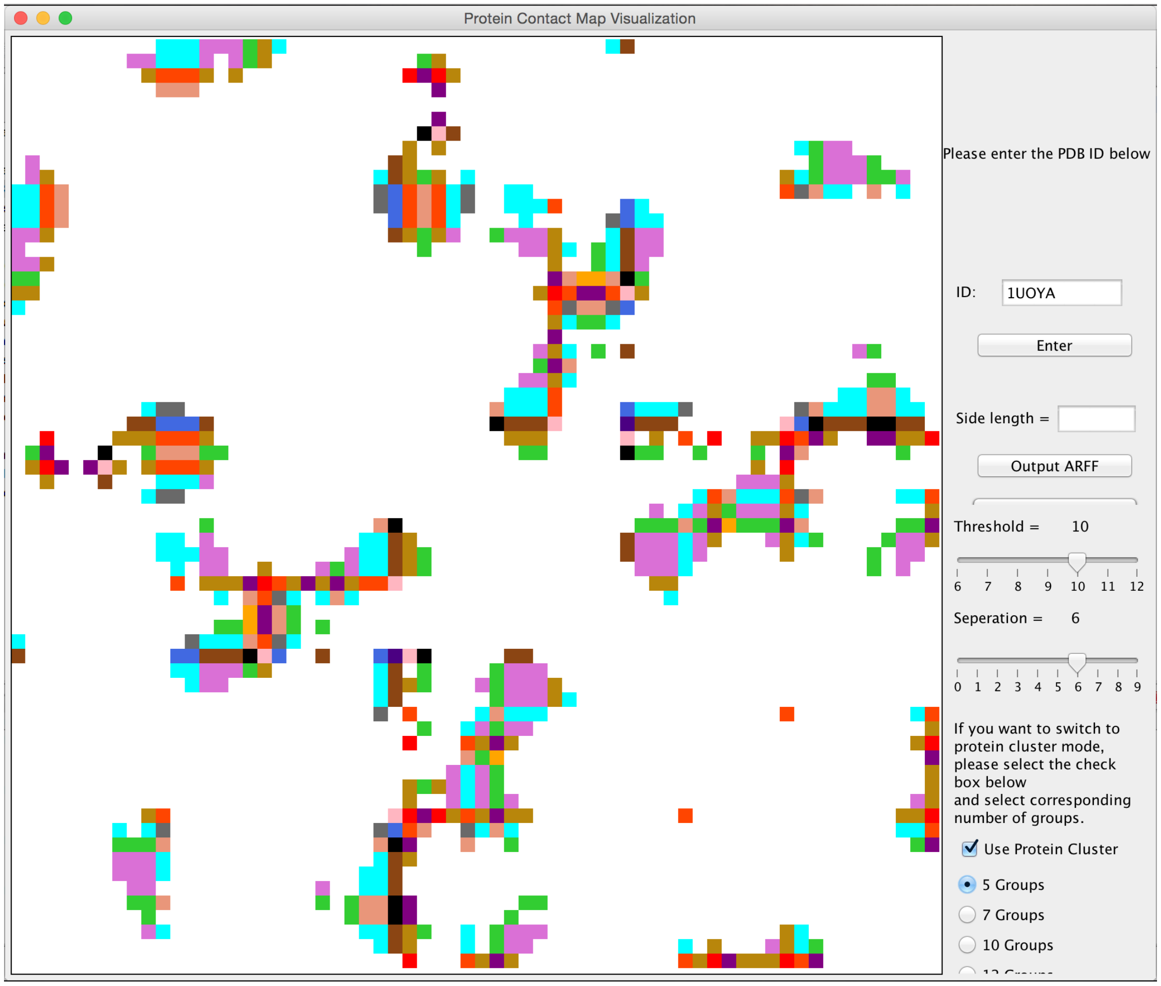
7 Groups

10 Groups

Figure 4.1: Protein contact map of sequence A in protein 1UOY; threshold = 8; separation 6; no clusters.

4.2 Amino Acid Clustering

To reduce the feature space, twenty amino acids are clustered using hierarchical clustering with complete linkage based on their physicochemical properties. The dendrogram is shown in **Figure 4.3**. Based on this dendrogram, the number of amino acid clusters can vary from 5 to 15; we used 10 in our work. In this way, the feature space


Protein Contact Map Visualization

Please enter the PDB ID below

ID: 1UOYA

Side length =

Threshold = 10

Seperation = 6

If you want to switch to protein cluster mode, please select the check box below and select corresponding number of groups.

[✓] Use Protein Cluster

[● 5 Groups]

7 Groups

10 Groups

Figure 4.2: Protein contact map of sequence A in protein 1UOY; threshold= 10; separation 6; 5 clusters.

is substantially reduced and our decision tree model is further generalized.

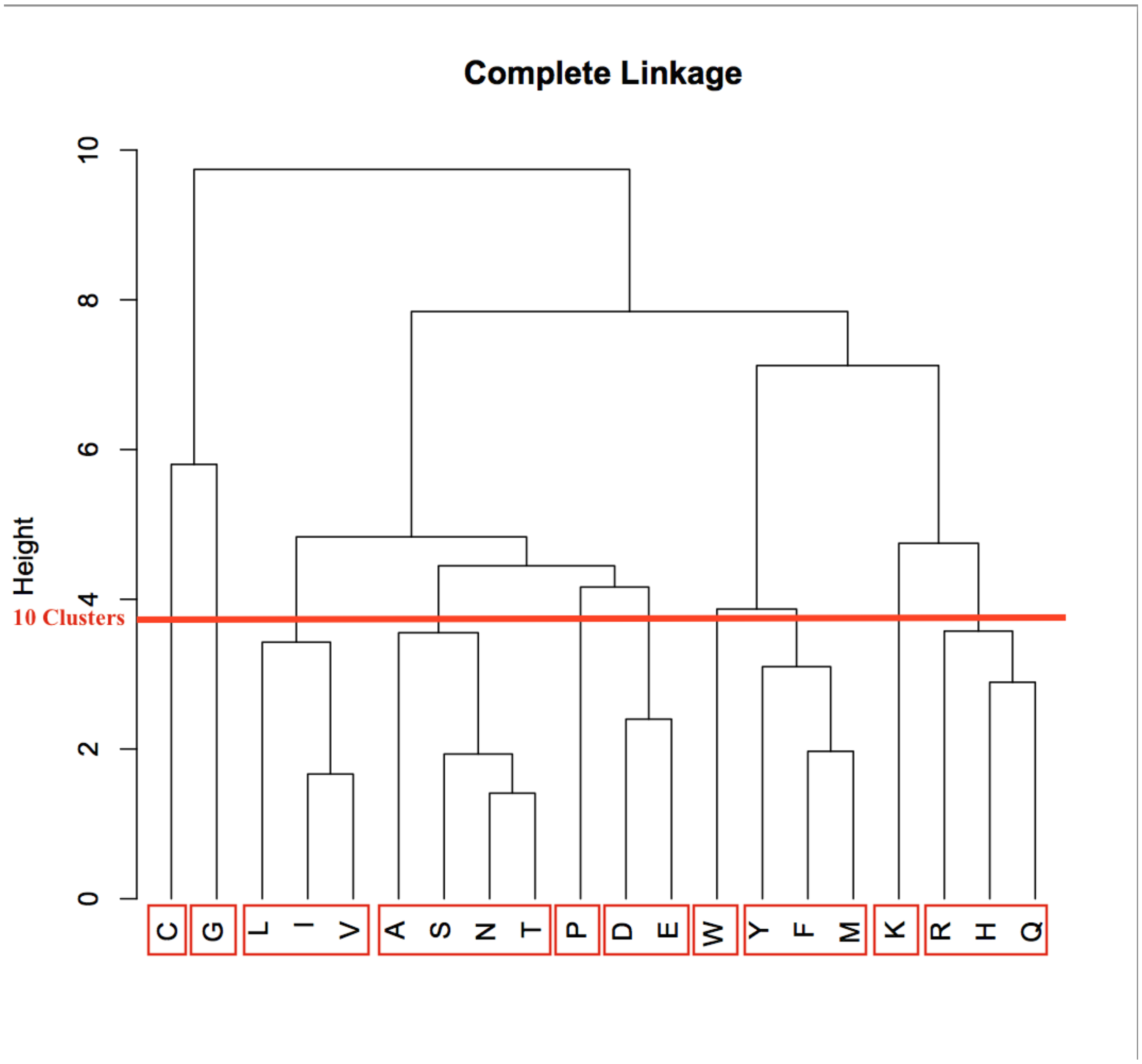


Figure 4.3: Twenty amino acids are clustered based on their physicochemical properties using hierarchical clustering with complete linkage. The dendrogram is the typical output from a hierarchical clustering method.

4.3 Performance Evaluation

4.3.1 Sample Size

To produce ROC curves, we sorted all proteins based on their AUC values, using a minNode of 25, and selected 105 of the top performing protein sequences from our

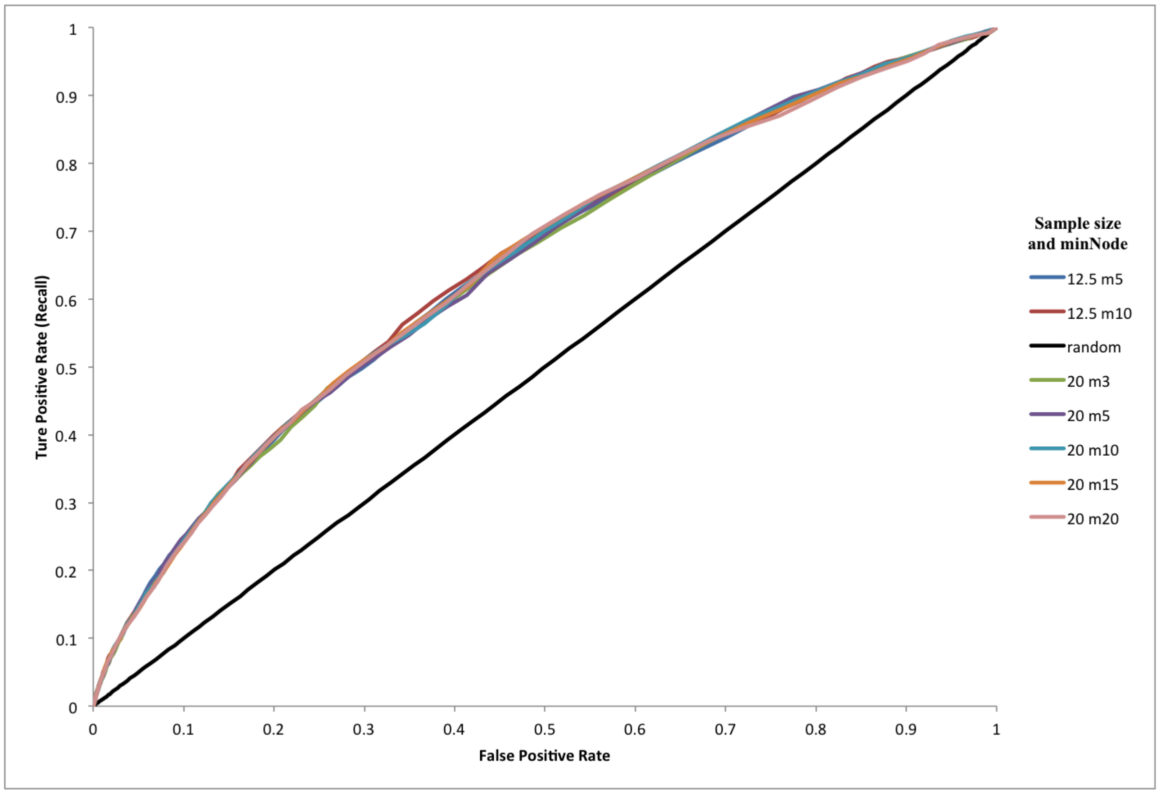


Figure 4.4: The ROC curves of 105 protein sequences over five minNode values with 20% sample size and two minNode values with 12.5% sample size. The black line is output from random.

data set. We then output ROC curves over six different minNode values, as shown in **Figure 4.4**. Proteins 2W6AA, 1N7SD, 1N7SB, 1JCDA did not have any contacts of 8Å, and therefore were not included in our results. The resulting ROC curves suggest that our model can predict protein contact maps much better than random.

As we discussed in **Section 3.4**, a resampling technique is used to balance the class distribution of our dataset. The minority class, which is the class that contains only contacts, ranges from 5% to 8% of the whole dataset. Using a sample size of 12.5% will give us a relatively uniform distribution over minority and majority class.

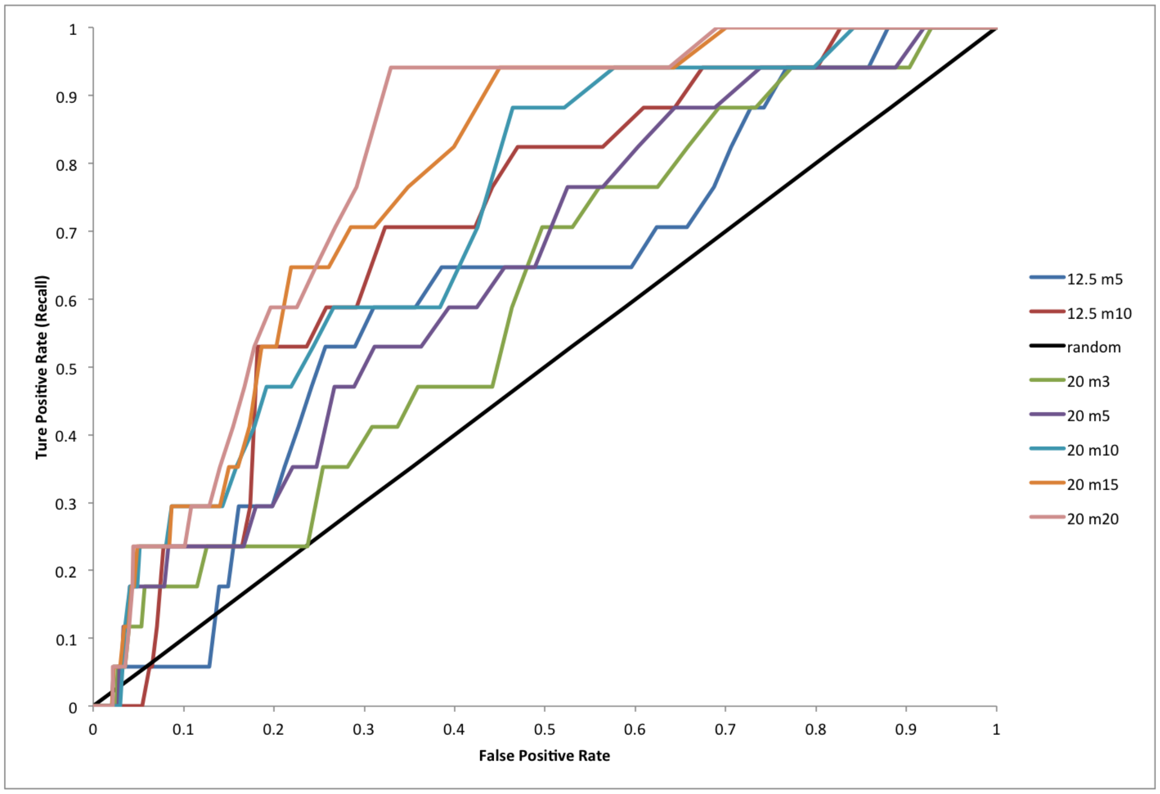


Figure 4.5: The ROC curves of protein 3H4NA over five minNode values with 20% sample size and two minNode values with 12.5% sample size. The black line is output from random.

On the other hand, a sample size of 20% will give us more data points, especially those from the majority class (non-contacts). Though it is not obvious from **Figure 4.4** which sample size had the most optimal results, we observed numerous cases where larger minNode values obtained better results. For example, the ROC curve of protein 3H4NA (**Figure 4.5**) indicates that sample size of 20% performs better than sample size of 12.5%, especially for high minNode values. However, there were cases where 12.5% performed the best as well.

Table 4.1 gives a better comparison between two sample sizes. Our model gave

Table 4.1: Table of AUC results for all proteins with valid AUC values aggregated over m values of sample size 20% and sample size of 12.5%.

	Sample Size					
	12.5 %		20%			
minNode	5	10	5	10	25	50
Min	0.470	0.485	0.481	0.474	0.495	0.528
Max	0.838	0.836	0.841	0.833	0.838	0.829
Average	0.658	0.663	0.660	0.665	0.667	0.670
Standard Deviation	0.079	0.078	0.082	0.079	0.076	0.0679

slightly larger average AUC values for sample size 20% with higher standard deviation for minNode = 5 and 10. Nevertheless, if we increase the minNode to 25, our model gave us the best average results, with lowest standard deviation. We measured the standard deviation of prediction performance over all proteins. Models with larger minNode values and sample size yielded more stable results. More importantly, larger minNode values and sample sizes rarely yielded predictions worse than random (AUC < 0.5). Hence, we selected a sample size of 20% for the majority of our tests.

4.3.2 minNode Value

Table 4.2: Table of AUC results for all proteins with valid AUC values aggregated over all m values of sample size 20%

	minNode								
AUC	3	5	10	15	20	25	**50**	75	100
Min	0.467	0.481	0.474	0.486	0.489	0.495	**0.528**	0.478	0.478
Max	0.839	0.841	0.833	0.831	0.833	0.838	**0.829**	0.813	0.810
Average	0.657	0.660	0.665	0.666	0.667	0.667	**0.670**	0.645	0.643
Standard Deviation	0.0816	0.0815	0.0795	0.0782	0.0774	0.0761	**0.0679**	0.073	0.072

Table 4.2 is a summary of AUC results for all proteins of sample size of 20%, except

those that have no positive predictions. Recall that trees with higher minNode are more general, and hence are more robust from overfitting. However, if the tree is too general, it cannot capture the details from the dataset. The result presented in **Table 4.2** demonstrates this claim. The average AUC increases from minNode 3 to 50, but starts to decrease when minNode is larger than 50. Though trees with lower minNodes (minNode = 3 or 5) have higher maximum AUC values, they resulted in the worst minimum AUC. In addition, their standard deviations are higher than those with higher minNodes (minNode = 25 or 50). Furthermore, when minNode = 50, all the predictions are better than random (AUC > 0.5). Therefore, from **Table 4.2** we can conclude that 50 is the most optimal value, because it has the highest average AUC values, but the lowest standard deviation.

Protein 4F14A and 3M0RA are among the top performing proteins. The ROC curves of 4F14A and 3M0RA over six different minNode values with sample size of 20% are shown in **Figure 4.6** and **Figure 4.7**, respectively. The overall performance for all six minNode values are similar, with the best performance observed when minNode is relatively small, especially when minNode = 3 or 5.

Figure 4.8 shows the ROC curve of protein 1UOYA with sample size 20%. Our model was slightly better than random guess on protein 1UOYA. In this worst-case scenario, it is clear that our model performed the best when minNode = 25. It is even more obvious in protein $3H4NA$, whose ROC curve is not smooth because of very few predictions obtained. This happens when a protein is under represented in this dataset. Clearly, from **Figure 4.9** we can conclude that our model performed

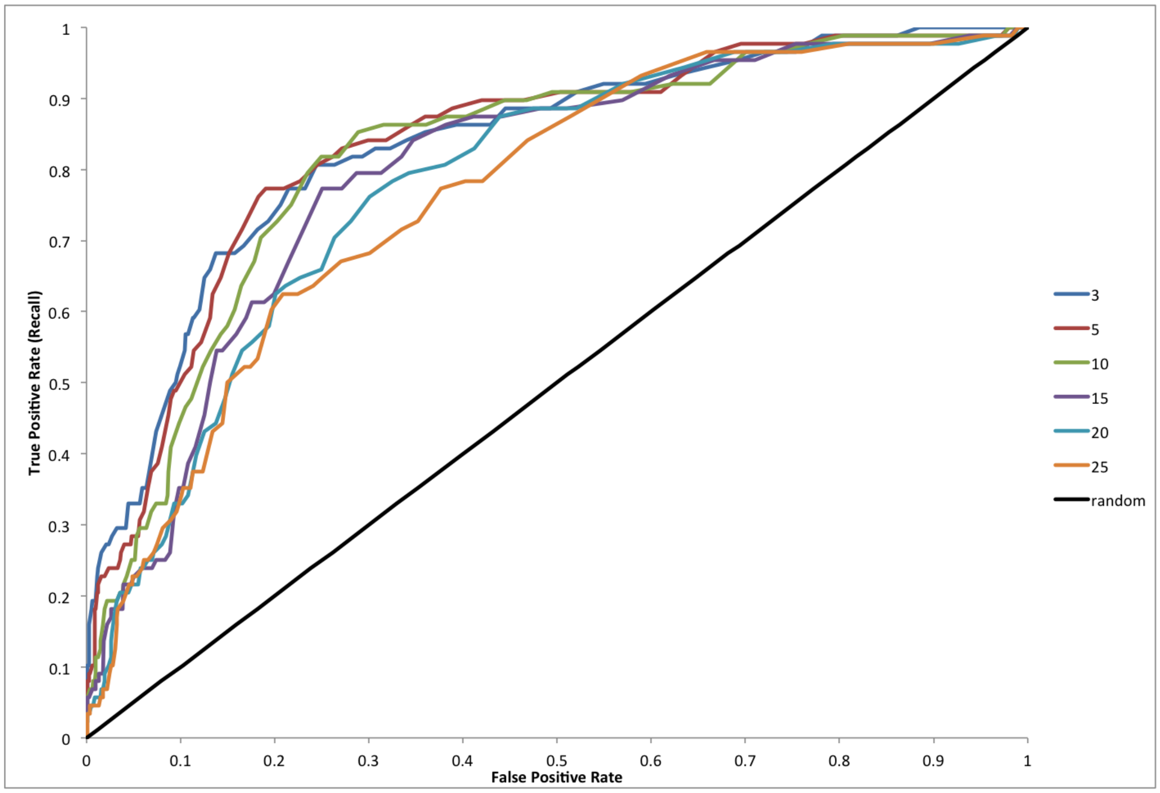


Figure 4.6: The ROC curve of protein 4F14A over six minNode values with 20% sample size. The black line is output from random.

the best when minNode = 25. Though a large minNode model cannot capture every nuance of a protein, it can be generalized over a wide variety of proteins. This explains why our model with larger minNode values didn't output better AUC for protein $4F14A$ and $3M0RA$, whose structure are rich in the dataset, but had better results for protein $1UOYA$, which is under represented in this dataset. The trend of minNode became extreme for protein $3H4NA$ (ROC curve shown in **Figure 4.5**), where the AUC value of sample size = 20% and minNode = 3 is even smaller than that of sample size = 12.5% and minNode = 10.

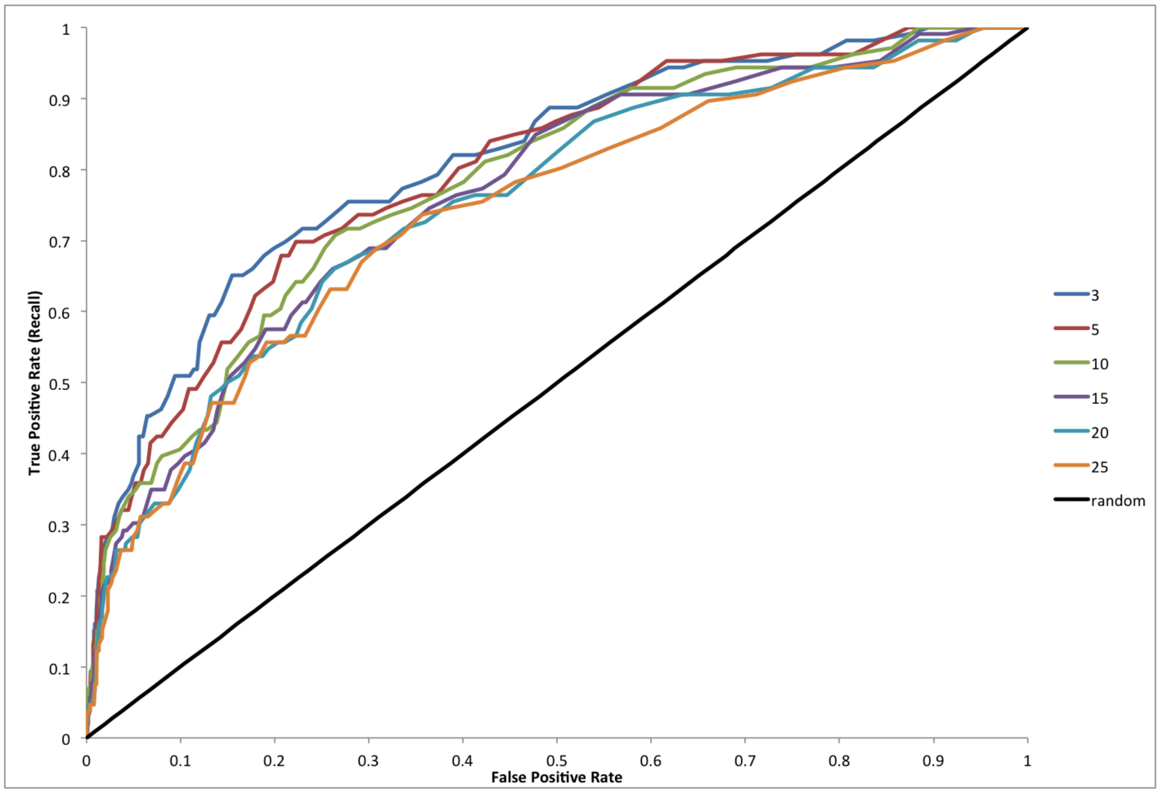


Figure 4.7: The ROC curve of protein 3M0RA over six minNode values with 20% sample size. The black line is output from random.

4.3.3 Overall Performance

Due to the linear relationship between the number of protein contacts and protein length, we calculate accuracy and recall for the first L, $L/2$ and $L/5$ predictions, as described in **Section 3.6**. **Table 4.3** is a summary of the range of accuracy and recall for the first L, $L/2$ and $L/5$ predictions, where L is the length of the protein in the test set. We output the range instead of the average value for accuracy and recall because the performance of our classifier varies on different protein sequences. We will further examine this issue in **Chapter 5**. The accuracy increases as we move from L to $L/5$,

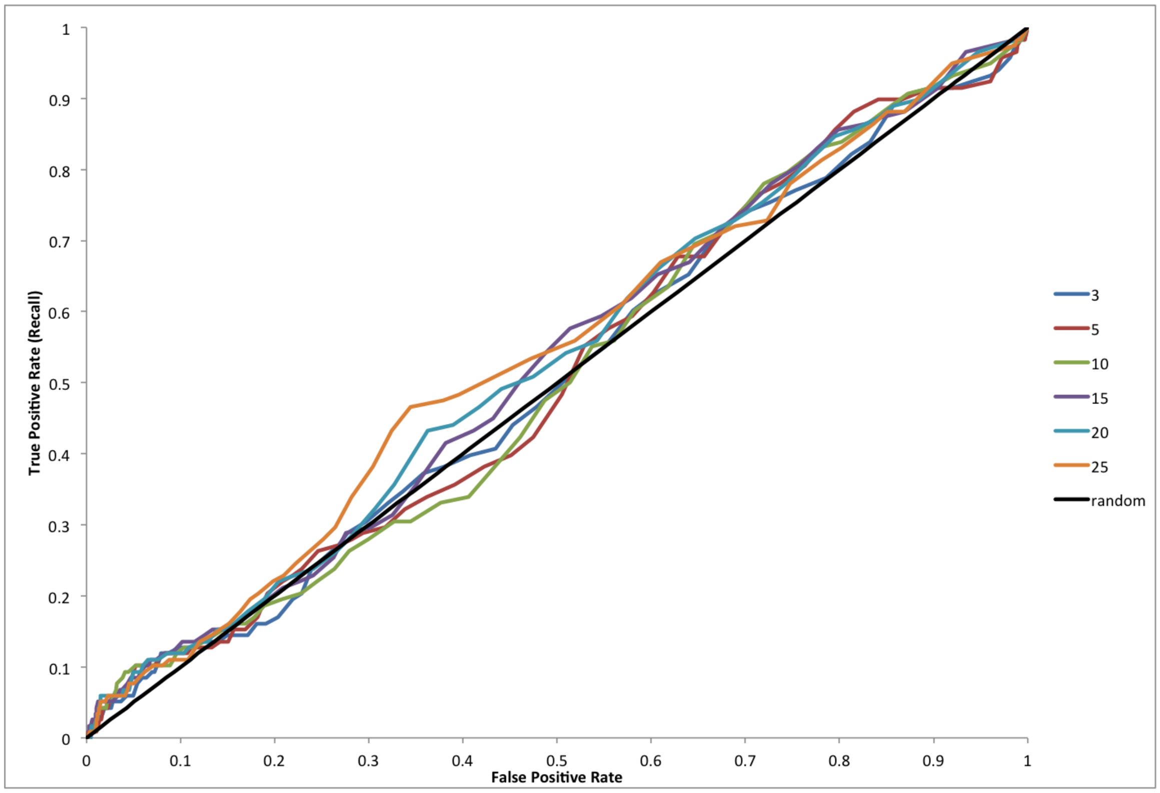


Figure 4.8: The ROC curve of protein 1UOYA over six minNode values with 20% sample size. The black line is output from random.

while the recall decreases. This is because the total number of predicted contacts decreases as we reduce the number of predictions, but the observed contacts stays the same. This leads to a smaller recall, but a larger accuracy. Furthermore, because the length of proteins in this dataset is relatively small, some of the proteins might not have enough contacts at $L/5$. As a result, for some proteins, results of $L/5$ is worse than that of $L/2$, e.g. protein $1F94A$ and protein $1ZUUA$ shown in **Table 4.4**.

To further analyze the performance of our model, we compared the performance based on each protein. **Table 4.4** shows the results from top ten proteins and **Ta-**

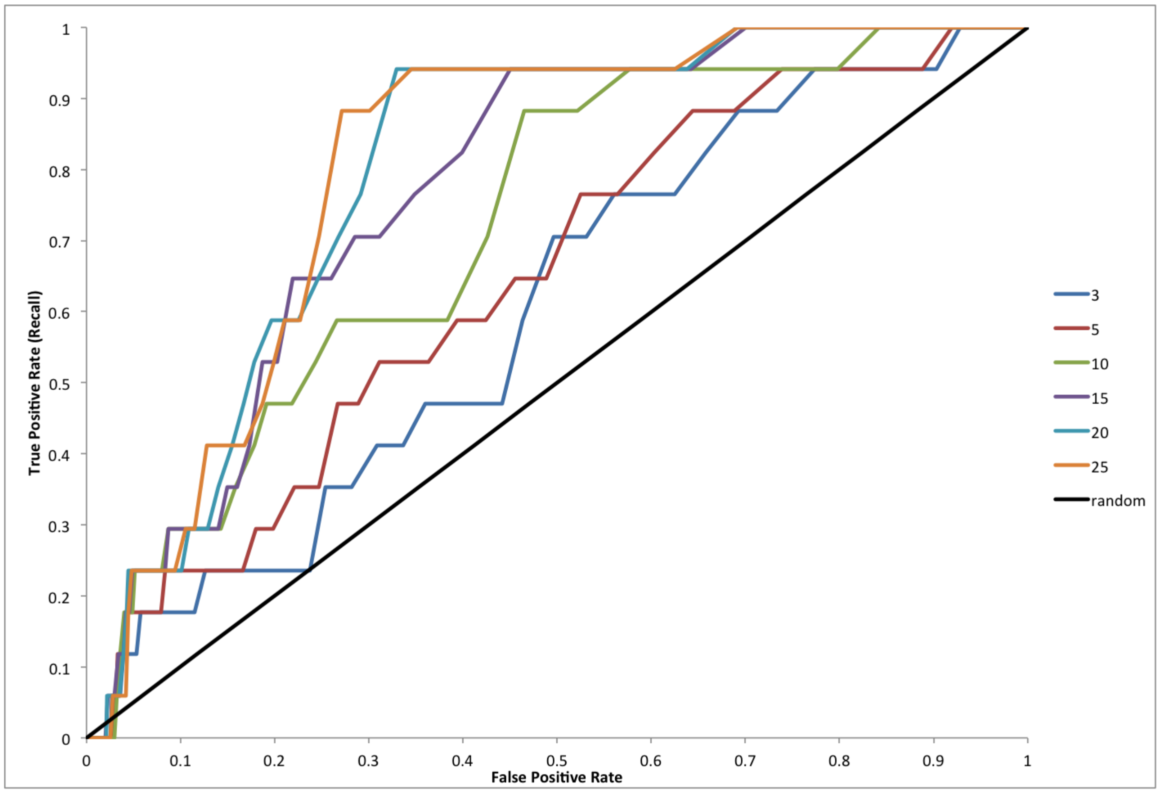


Figure 4.9: The ROC curve of protein 3H4NA over six minNode values with 20% sample size. The black line is output from random.

ble 4.5 reveals results from the ten worst proteins. Both **Table 4.4** and **Table 4.5** demonstrate that our classifier perform significantly different from protein to protein. We will discuss this issue in more details in **Chapter 5**.

Table 4.3: The aggregated results of all 105 proteins with valid AUC values for L, $L/2$ and $L/5$ with minNode = 20 and sample size 20%.

	L	L/2	L/5
Acc	0.013 - 0.38	0.028 - 0.625	0.0625 - 0.692
Rec	0.02 - 0.33	0.018 - 0.18	0.0092 - 0.133

Table 4.4: Performance of ten best proteins on L, $L/2$ and $L/5$, with sample size of 20% and minNode 25.

				L		L/2		L/5	
Protein ID	**Length**	**Alpha helices**	**Beta sheets**	**Accuracy**	**Recall**	**Accuracy**	**Recall**	**Accuracy**	**Recall**
3M0RA	51	35	0	0.483	0.264	0.520	0.142	0.455	0.047
3AWXB	72	21	24	0.403	0.271	0.464	0.159	0.429	0.056
1ZUUA	77	18	32	0.375	0.233	0.3913	0.133	0.182	0.022
3U23A	74	30	23	0.375	0.25	0.357	0.119	0.364	0.048
4F14A	50	47	0	0.339	0.216	0.261	0.091	0.364	0.045
2HLRA	78	14	25	0.299	0.187	0.455	0.141	0.538	0.065
2YEOA	77	45	0	0.277	0.140	0.344	0.085	0.385	0.039
2CS7A	78	17	12	0.273	0.333	0.167	0.222	0.182	0.044
1F94A	63	0	34	0.270	0.167	0.355	0.108	0.25	0.029
1G6XA	58	12	15	0.259	0.147	0.310	0.088	0.182	0.020

Table 4.5: Performance of ten worst proteins on L, $L/2$ and $L/5$, with sample size of 20% and minNode 25.

				L		**L/2**		**L/5**	
Protein ID	**Length**	**Alpha helices**	**Beta sheets**	**Accuracy**	**Recall**	**Accuracy**	**Recall**	**Accuracy**	**Recall**
3H87C	66	0	34	0.0139	0.111	0	0	0	0
3T47B	77	19	36	0.0143	0.0278	0	0	0	0
2CZSB	68	52	0	0.0145	0.0294	0.167	0.0294	0.167	0.0294
2G7OA	52	17	12	0.0147	0.1	0	0.1	0	0
1NKDA	64	9	23	0.0170	0.0385	0	0	0	0
2RKLA	72	3	37	0.0189	0.0385	INVALID	0	INVALID	0
3OMYA	56	3	31	0.0196	0.0769	0	0	0	0
1Z0NB	75	64	0	0.0375	0.0246	0	0	0	0
1GVDA	52	29	0	0.0385	0.0833	0.0556	0.0417	0	0
2WUJA	63	22	13	0.04	0.222	0	0.222	0	0.020

Chapter 5

Discussion

We experimentally determined each parameter of the decision trees we used in our work. We presented our result in the previous chapter, and in this chapter, we will explain why we selected these parameters. Furthermore, in the previous chapter, we showed **Table 4.4** and **Table 4.5**. We discovered that proteins with different secondary structure have different performance. In this chapter, we will further explore this idea.

5.1 Model Parameter

Each decision tree was induced using the $J48$ implementation in the Weka machine learning library. To control the complexity of each tree, we used $\mathbf{C = 0.5}$, where C denotes the expected *cost complexity* (in **Section 1.3.1**). Because contacts represent

Table 5.1: Table of AUC results for selected proteins over all m values of sample size 20%.

	minNode								
Protein ID	3	5	10	15	20	25	50	75	100
3M0RA	**0.817**	0.805	0.786	0.772	0.761	0.751	0.721	0.687	0.681
4F14A	**0.839**	**0.839**	0.825	0.803	0.789	0.776	0.736	0.709	0.699
2UX9A	0.644	0.651	**0.658**	0.650	0.646	0.645	0.632	0.617	0.600
1WM3A	0.751	0.749	0.748	**0.758**	0.747	0.739	0.702	0.659	0.644
1UOYA	0.516	0.519	0.517	0.538	0.543	0.552	**0.571**	0.566	0.564
3H4NA	0.613	0.657	0.733	0.786	0.808	0.815	**0.829**	0.805	0.774
1TG0A	0.600	0.603	0.608	0.607	0.618	0.614	0.630	**0.632**	0.631
1B67A	0.646	0.667	0.689	0.691	0.704	0.708	0.754	0.672	**0.717**

5% to 8% of the overall dataset, the resampling technique gives a more uniform distribution over two classes (contacts and non-contacts). This is particularly true for a sample size of 12.5%. With a sample size of 20%, more data from non-contact class will be drawn from the data without replacement. Though it seems that sample size of 12.5% will output better AUC values over sample size of 20%, **Table 4.1** has shown that our model output larger average AUC values for sample size 20% with minNode = 5 and 10. This implies that information from non-contact data is important to our model. A sample size of 20% increased the proportion of contacts in the sample, while kept as much information from non-contact class as possible at the same time.

We did more experiments with sample size of 20%, and we found that if we increased the minNode to 25 and 50, our model gave us the best average results, with the lowest standard deviation. **Table 4.2** shows the result from different minNode values with the same sample size (20%). As minNode values increase, the average AUC increases and the standard deviation decreases. This indicates that our model performed the best at minNode = 50. A decision tree with large minNode value is more general than that with small minNode value. A more generalized decision tree may not perform the best on some proteins, such as 1WM3A, 4F14A and 3M0RA, whose AUC at minNode 50 is the lowest among all other minNode values shown in **Table 5.1**, but it will give the best average AUC, as shown in **Table 4.2**.

Our dataset was prepared such that each protein was at most 25% similar to other proteins in the data. But the structures of some proteins might be better represented in this dataset, due to amino acid clustering. In this case, decision trees with smaller minNode values will perform better on these proteins (e.g. 1WM3A, 4F14A and 3M0RA), because they can capture more details in protein structures. Nevertheless, these decision trees may suffer from overfitting problem; that is, their performance is biased toward information contained in the training sets and therefore varies a lot. Even though these decision trees have great performance on some proteins, they may perform extremely poorly on other proteins whose structures are not or are under represented in the training set, such as 1UOYA or 3H4NA, as shown in **Table 5.1**. Particularly, our model improved a lot as minNode value increases on protein 3H4NA, indicating that the structure of 3H4NA is quite unique in this dataset.

5.2 Protein Secondary Structures

In the previous chapter, we have shown the accuracy and recall for both ten top performing proteins (**Table 4.4**) and ten worst performing proteins (**Table 4.5**). In the top performing proteins, there are 3 proteins that contain only alpha helices, and 1

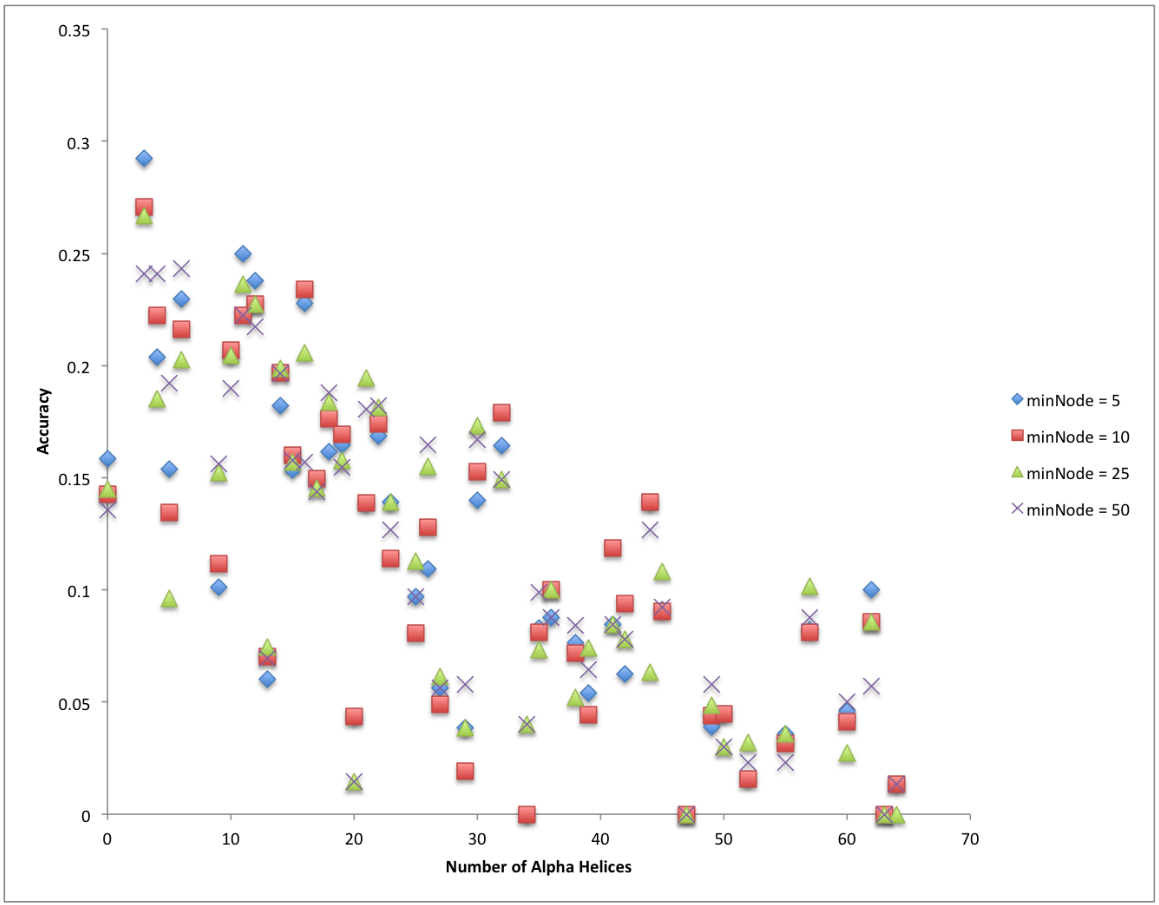


Figure 5.1: Number of alpha helices versus accuracy for the full length of protein at threshold of 8 with 20% sample size and various minNode values.

protein that contains only beta sheets. For proteins containing both alpha helices and beta sheets, 4 of them have more beta sheets than alpha helices. Among all bottom performing proteins, 3 of them only have alpha helices, 1 only contain beta sheets, and 6 contain both alpha helices and beta sheets – 2 out of 6 have more alpha helices than beta sheets. Though it is not clear from **Table 4.4** and **Table 4.5** whether there is relationship between the performance of our model and proteins' secondary structure, we found positive relationship between the number of beta sheets and accuracy and negative relationship between the number of alpha helices and accuracy

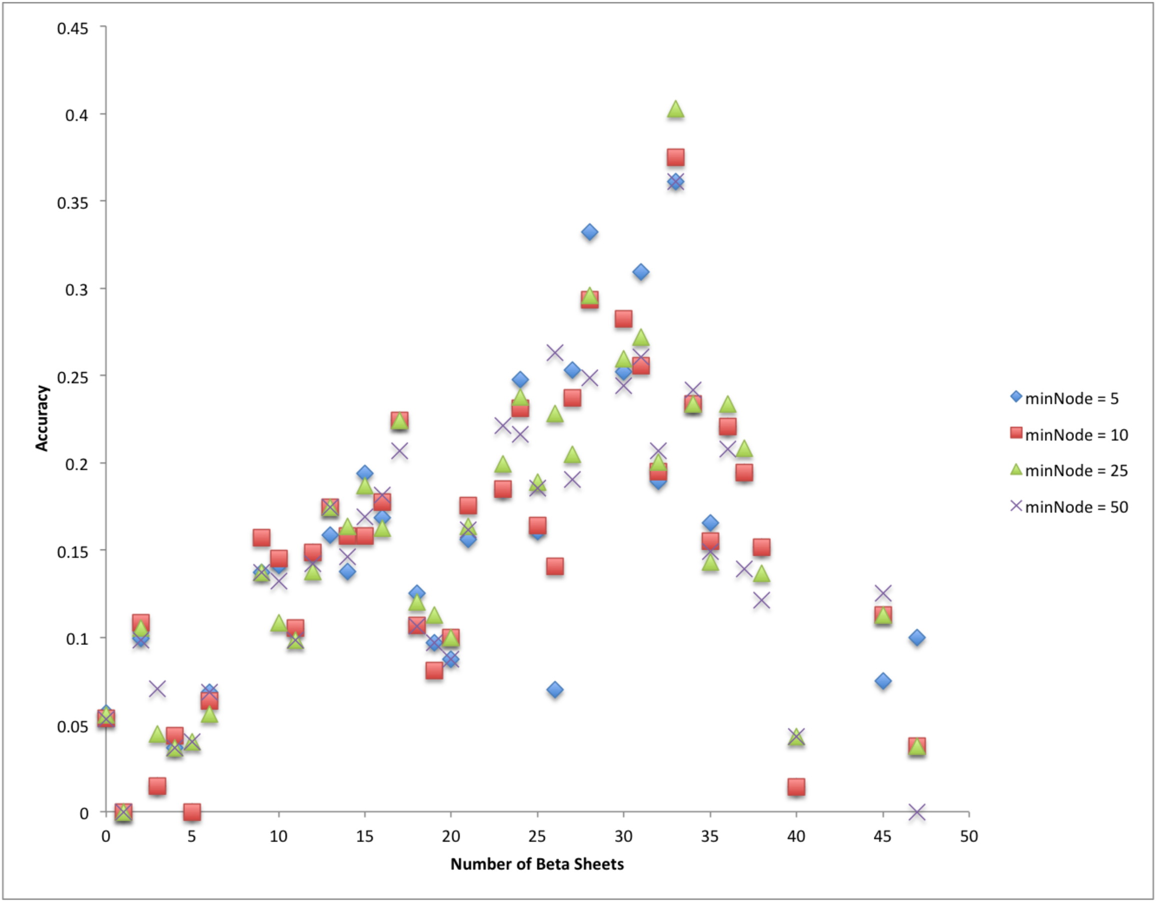


Figure 5.2: Number of beta sheets versus accuracy for the full length of protein at threshold of 8 with 20% sample size and various minNode values.

at threshold of 8 with sample size 20%. From **Figure 5.1** we can see that as the number of alpha helices increases, the accuracy decreases; while in **Figure 5.2**, as the number of beta sheets increases, the accuracy increases as well. Hence we can conclude that our model performed better on beta sheets than alpha helices.

Figure 5.3 shows that at threshold 8, proteins with more alpha helices have less contacts than proteins with more beta sheets. We believe that the insufficient contacts led to poor performance of our model on alpha-helices rich proteins. **Figure**

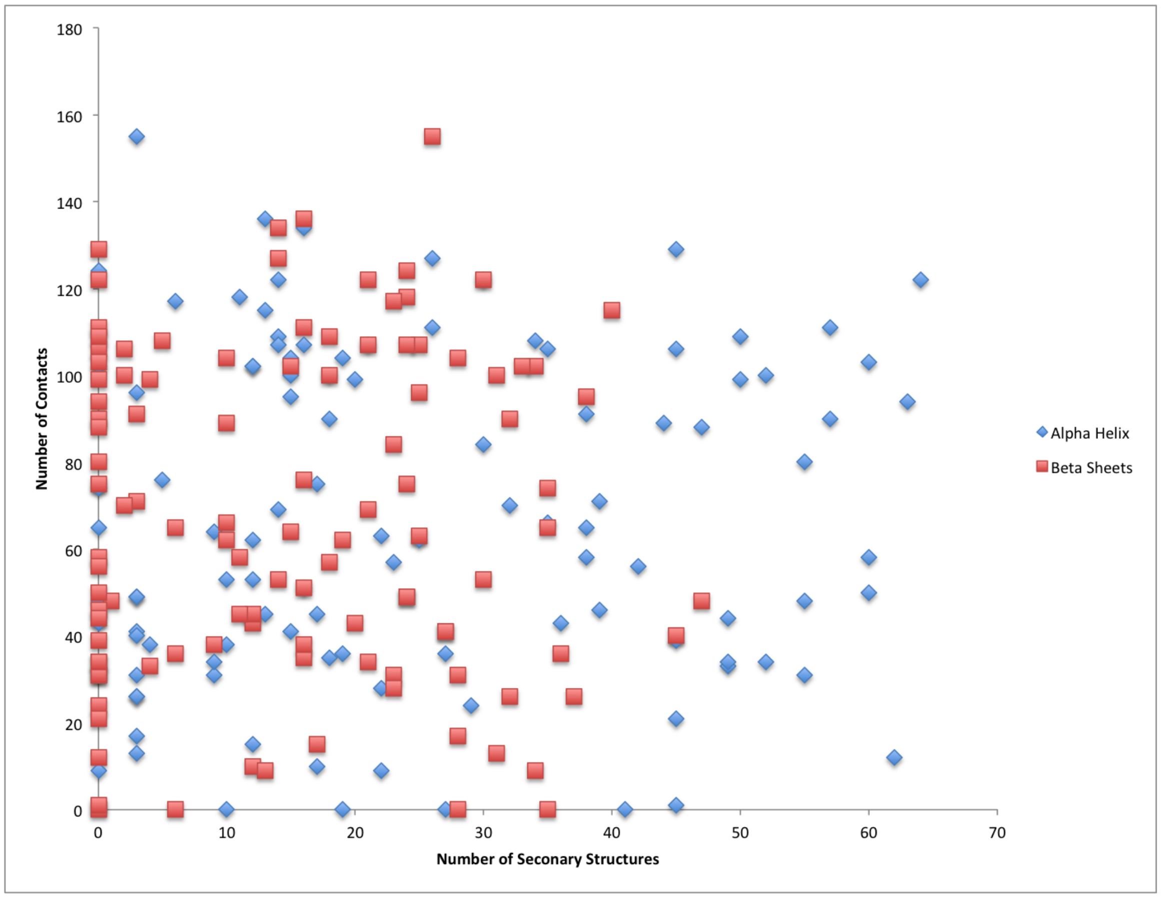


Figure 5.3: Number of contacts versus number of residues in secondary structures at threshold of 8 with 20% sample size.

5.4 shows that, by increasing threshold to 10, the number of contacts increases substantially for all proteins. Our model performed significantly better on proteins with more alpha helices at threshold of 10, as shown in **Figure 5.5**. However, on the other hand, the performance did not substantially improve for proteins with more beta sheets. **Figure 5.5** shows significant improvement of accuracy for proteins with more alpha helices when we increases the threshold to 10, while **Figure 5.6** indicates that our model didn't improve a lot on proteins with more beta sheets.

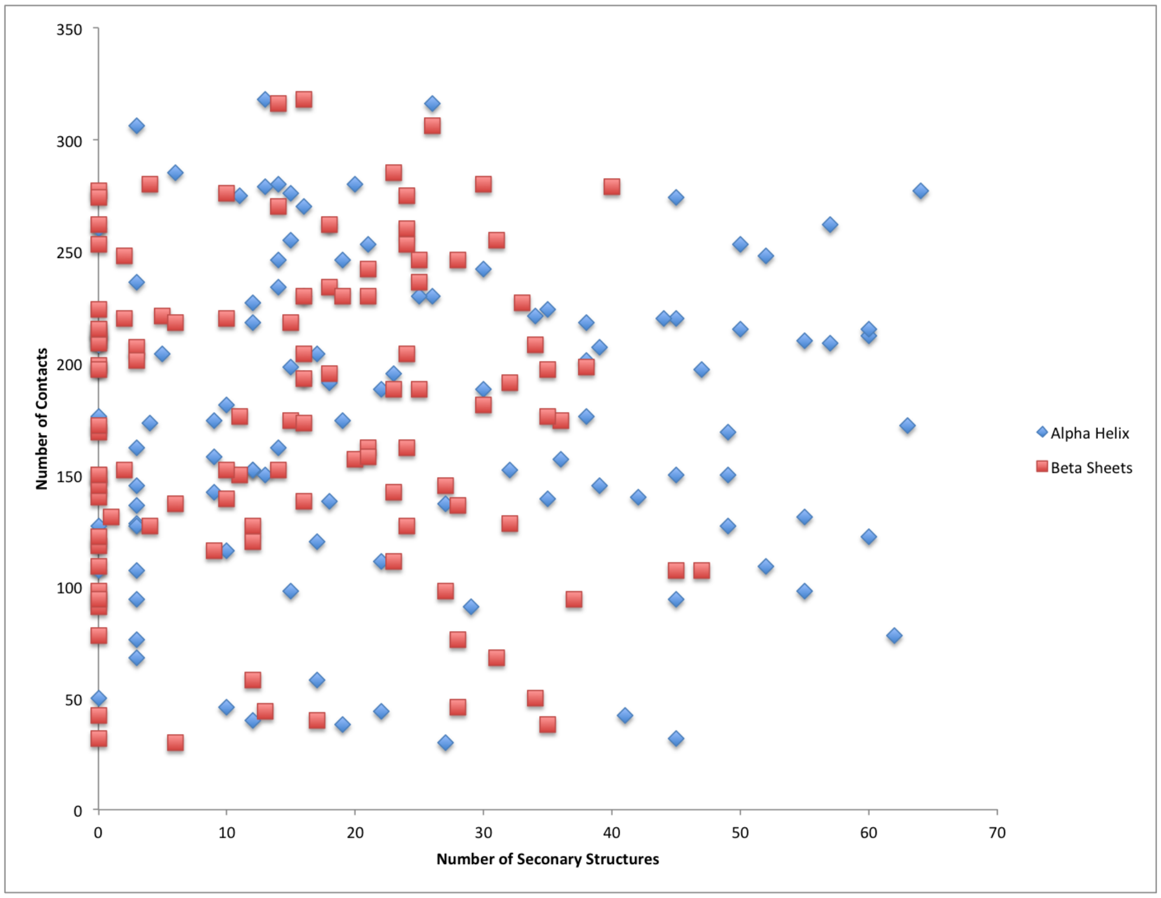


Figure 5.4: Number of contacts versus number of residues in secondary structures at threshold of 10 with 20% sample size.

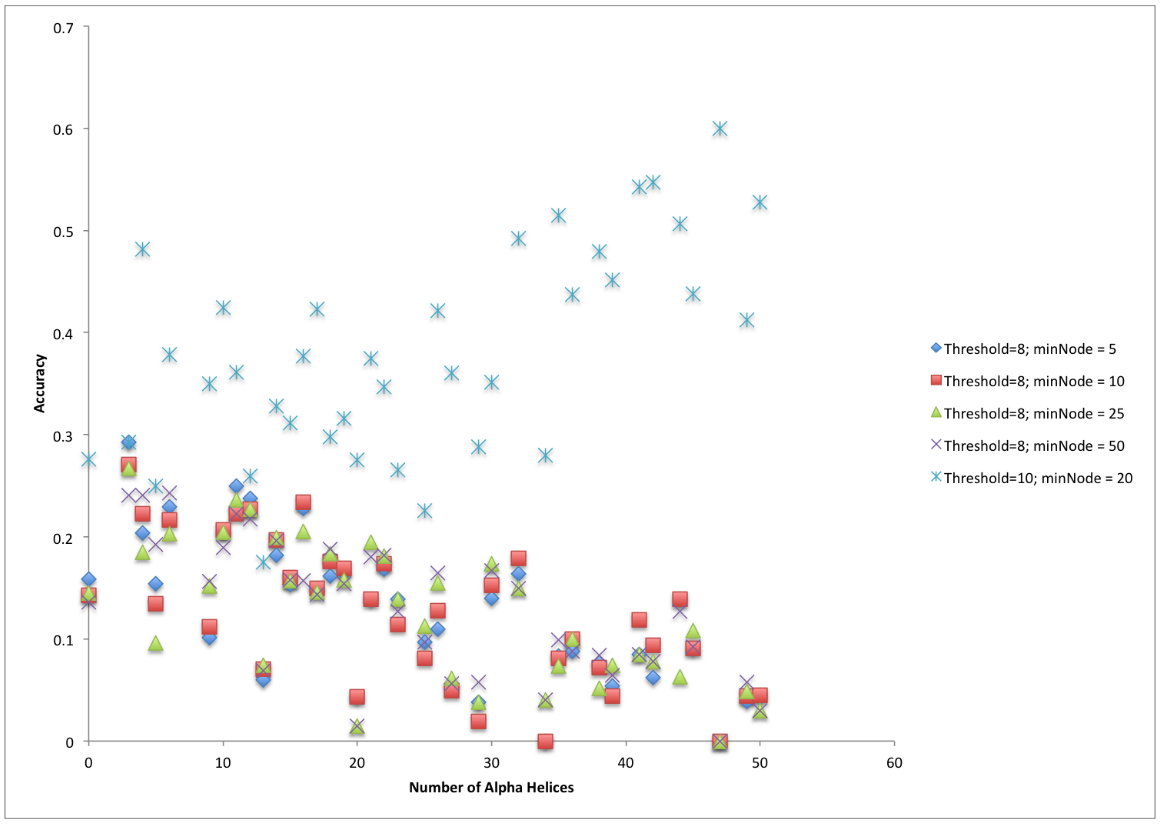


Figure 5.5: Number of alpha helices versus accuracy for the full length of protein at threshold of 10 with 20% sample size and minNode of 20 comparing with that at threshold of 8 with 20% sample size and various minNode values.

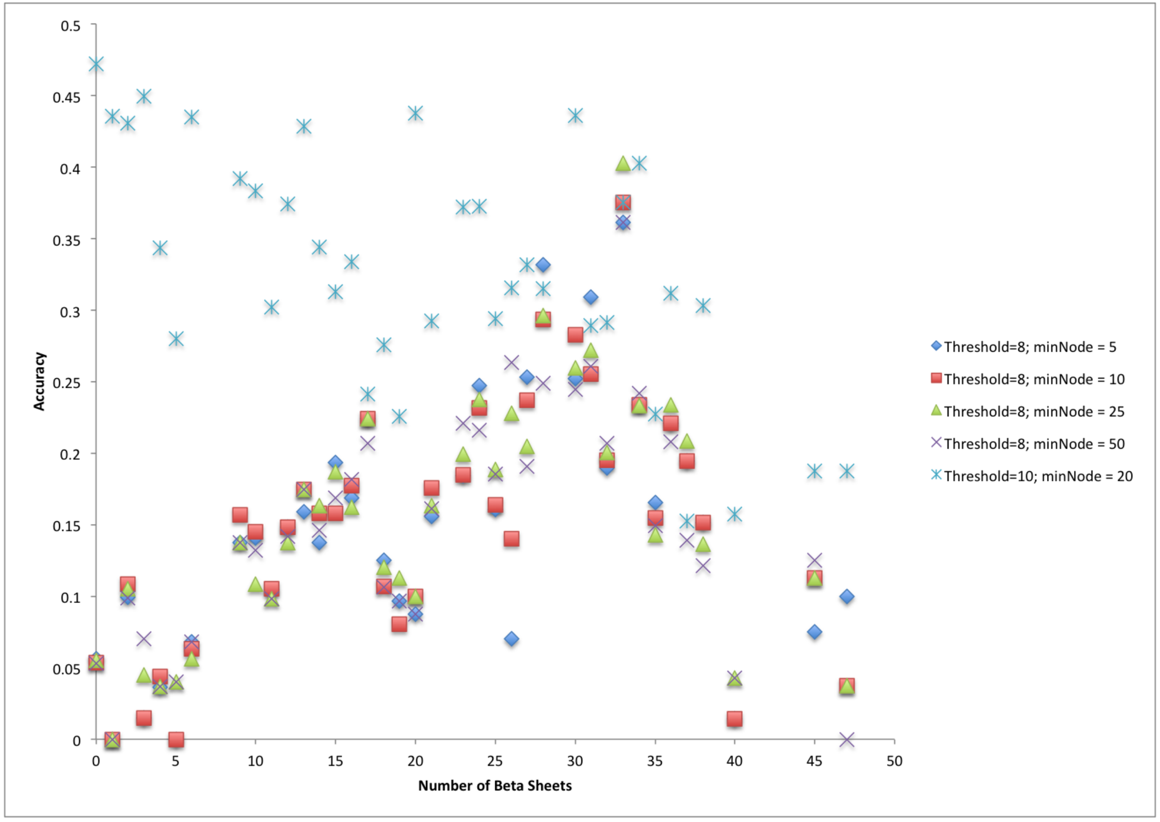


Figure 5.6: Number of beta sheets versus accuracy for the full length of protein at threshold of 10 with 20% sample size and minNode of 20 comparing with that at threshold of 8 with 20% sample size and various minNode values.

Chapter 6

Conclusion and Future Work

Protein contact maps are concise representations of proteins' three-dimensional structure. Protein contact map prediction has been a great challenge to biologists and computer scientists. In this thesis, we introduced a novel ensemble method by bagging decision trees to predict protein contact map. In **Chapter 3**, we introduced our model, bagging Quinlan's C4.5 decision tree [10] with majority vote and a probabilistic model. We constructed our model based on Weka's $J48$ decision tree class. We generalize the feature space by reducing the number of amino acid from twenty to ten using hierarchical clustering with complete linkage. In **Chapter 4**, we showed the results for different minNode values and sample sizes, and evaluated the overall accuracy and recall for the first L, $L/2$ and $L/5$ predictions, where L is the length of a given protein. We also produced the overall ROC curves along with the individual ROC curves and AUC values for the top performing and worst performing proteins. From the results we concluded that our model performed the best with larger minNode values and sample size. Finally in **Chapter 5**, we analyzed and explained the parameters used in our model. We also discussed the potential for improvement by integrating protein secondary structure to our model.

6.1 Future Work

Although we concluded that minNode = 50 is the optimal parameter for our model, it is still not clear whether the results from minNode of 50 is the maximum or not. We need to perform more experiments with larger minNode values in the future. In the previous section we found that protein's secondary structure has great impact on the accuracy of our model. Therefore, we plan to integrate features of protein's secondary structure into our model. However, we don't have prior knowledge of protein's secondary structure; the only information we have is the protein sequence. So it is also essential for us to include a secondary structure predictor to our model for future reference. Our preliminary results showed that our model obtains a significant improvement in accuracy of alpha-rich proteins. This is shown in **Figure 5.5**. Increasing the threshold from 8Å to 10Å allowed us to observe a positive trend in accuracy with respect to the number of residues in alpha helices. However, from **Figure 5.6** we found that the performance on beta sheets was not improved significantly by increasing threshold. Also, from **Figure 5.2** we saw a decrease in accuracy as the number of beta sheets increases over 35. Both figure implies that our model didn't perform well on proteins with a large number of beta sheets, and this cannot be solved by increasing threshold. Hence we need to find a way to accommodate this issue as well.

References

[1] M. Punta and B. Rost, "PROFcon: novel prediction of long-range contacts." Bioinformatics (Oxford, England), **21 13**, 2960 (2005).

[2] P. W. Rose, C. Bi, W. F. Bluhm, C. H. Christie, D. Dimitropoulos, S. Dutta, R. K. Green, D. S. Goodsell, A. Prlic, M. Quesada, G. B. Quinn, a. G. Ramos, J. D. Westbrook, J. Young, C. Zardecki, H. M. Berman, and P. E. Bourne, "The RCSB Protein Data Bank: new resources for research and education," Nucleic Acids Research, **41 D1**, D475 (2014).

[3] I. Friedberg, L. Jaroszewski, Y. Ye, and A. Godzik, "The interplay of fold recognition and experimental structure determination in structural genomics." Current opinion in structural biology, **14 3**, 307 (2004).

[4] W. Ding, J. Xie, D. Dai, H. Zhang, H. Xie, and W. Zhang, "CNNcon: improved protein contact maps prediction using cascaded neural networks." PloS one, **8 4**, e61533 (2013).

[5] N. M. Luscombe, D. Greenbaum, and M. Gerstein, "What is bioinformatics ?

An introduction and overview," Yearbook of Medical Informatics, pp. 83–100 (2001).

[6] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and L. Dj, "Gapped BLAST and PSI- BLAST: a new generation of protein database search programs," Nucleic acids Res, **25 17**, 3389 (1997).

[7] Y. Zhang, "Progress and challenges in protein structure prediction." Current opinion in structural biology, **18 3**, 342 (2008).

[8] N. Nilsson, *Introduction to machine learning* (1998).

[9] J. Han, M. Kamber, and J. Pei, *Data Mining*, 3 ed. (Morgan Kaufmann, 2012).

[10] J. Quinlan, "Induction of decision trees," Machine learning, pp. 81–106 (1986).

[11] P.-n. Tan and M. Steinbach, *Introduction to Data Mining*, 1 ed. (Addison-Wesley, 2006).

[12] T. G. Dietterich, "Ensemble Methods in Machine Learning," Lecture Notes in Computer Science, **1857**, 1 (2000).

[13] M. Vendruscolo, E. Kussell, and E. Domany, "Recovery of protein structure from contact maps." Folding & design, **2 5**, 295 (1997).

[14] C. Santiesteban-Toca and J. Aguilar-Ruiz, "DTP: decision tree-based predictor of protein contact map," Modern Approaches in Applied . . . , pp. 367–375 (2011).

[15] J. Cheng and P. Baldi, "Improved residue contact prediction using support vector machines and a large feature set." BMC bioinformatics, **8 1**, 113 (2007).

[16] A. Andreeva, D. Howorth, S. E. Brenner, T. J. P. Hubbard, C. Chothia, and A. G. Murzin, "SCOP database in 2004: refinements integrate structure and sequence family data." Nucleic acids research, **32 Database issue**, D226 (2004).

[17] C. Ren and B. R. King, "Predicting Protein Contact Maps by Bagging Decision Trees," Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, pp. 649–650 (2014).

[18] S. Griep and U. Hobohm, "PDBselect 1992-2009 and PDBfilter-select." Nucleic acids research, **38 Database issue**, D318 (2010).

[19] A. Sharma, K. K. Paliwal, A. Dehzangi, J. Lyons, S. Imoto, and S. Miyano, "A strategy to select suitable physicochemical attributes of amino acids for protein fold recognition." BMC bioinformatics, **14 1**, 233 (2013).

Appendix

Amino Acid Clustering

We selected a number of amino acids' physicochemical properties that play important roles in protein folding problem to cluster amino acid. The list of physicochemical properties includes hydrophobicity (H), polarity (P), van der Waals volume (W), reverse turn (T), size (S), molecular weight (M), volume (V) and average accessible surface area (AS). To capture the information from protein secondary structures, we also include the frequency of coil (C), the frequency of alpha helix (AH), the frequency of beta sheets (BS), and the frequency of each amino acid (AA). Because the scale of each entry is different, all values are standardize by normal distribution $\mathcal{N}(0,1)$. **Table 1** is a summary of all the physicochemical properties and frequencies of protein secondary structures used to cluster amino acids, and **Table 2** is a reflection of amino acid abbreviation. The Euclidean distance is then calculated and the dendrogram of hierarchical clustering is produced based on **Table 1**.

Amino Acid	H	P	S	T	S	M	V	AS	C	AH	BS	AA
A	-0.47	-0.62	-1.42	-0.46	-1.38	-1.55	-1.23	-0.79	-0.54	-0.66	-0.54	1.73
C	0.09	-0.55	-0.69	-0.36	-1.11	-0.51	-0.68	-1.3	2.51	2.71	3.05	-1.67
D	-0.65	1.65	-0.51	1.15	-1.38	-0.12	-0.7	0.58	-0.15	-0.19	-0.22	0.45
E	-0.64	1.66	0.01	0.1	-0.03	0.33	-0.02	0.89	-1.22	-0.89	-0.61	0.56
F	1.24	-0.6	1.10	-0.91	0.78	0.92	1.12	-0.88	-0.77	-0.68	-0.61	-0.51
G	-1.12	-0.62	-1.94	1.72	-2.46	-2	-1.89	-0.93	0.81	0.84	0.82	1.38
H	-0.47	1.73	0.46	-0.68	0.51	0.59	0.28	0.16	0.78	0.49	0.74	-1.28
I	1.48	-0.61	0.12	-1.11	0.24	-0.19	0.63	-1	0.51	0.50	0.24	0.24
K	0.18	1.64	0.52	0.02	1.05	0.3	0.81	2.34	1.98	1.99	1.58	0.46
L	0.64	-0.61	0.12	-0.93	0.24	-0.18	0.63	-0.8	-1.47	-1.22	-1.34	1.50
M	0.22	-0.56	0.34	-1.36	0.51	0.4	0.49	-0.55	-0.44	-0.32	-0.01	-1.29
N	-1.14	-0.47	-0.42	0.82	-0.03	-0.15	-0.65	0.55	-0.48	-0.46	-0.40	-0.15
P	1.15	-0.55	-0.54	2.4	0.24	-0.71	-1.2	0.2	-1.06	-0.72	-0.92	-0.13
Q	-1.21	-0.46	0.1	0.07	0.51	0.3	0.02	0.91	0.14	0.07	-0.02	-0.61
R	-0.48	1.75	1.22	-0.18	1.33	1.21	0.93	1.99	-0.68	-0.82	-0.69	-0.17
S	-1.15	-0.54	-1.12	0.92	-1.11	-1.03	-1.21	-0.2	-0.29	-0.53	-0.38	0.46
T	-1.15	-0.54	-0.6	0.22	-0.03	-0.58	-0.54	-0.07	0.14	0.06	-0.20	0.41

V	0.39	-0.61	-0.39	-1.21	-0.03	-0.64	0	-0.96	0.29	0.10	0.14	0.89
W	2	-0.52	2.23	-0.48	1.05	2.18	2	-0.5	-0.52	-0.98	-1.08	-1.64
Y	1.07	-0.55	1.4	0.25	1.05	1.43	1.21	0.35	0.43	0.73	0.47	-0.66

Table 1: This table contains values for each physicochemical properties and frequencies of protein secondary structures used for amino acid clustering. The entries for physicochemical properties are raw values, while the entries for frequencies are already standardized.

Table 2: This table reveals the abbreviation of each amino acid.

Amino Acid	Abbreviation
Aspartate	D
Glutamate	E
Phenylalanine	F
Glycine	G
Alanine	A
Cysteine	C
Leucine	L
Methionine	M
Asparagine	N
Histidine	H
Isoleucine	I
Lysine	K
Threonine	T
Tryptophan	W
Valine	V
Glutamine	Q
Proline	P
Serine	S
Arginine	R
Tyrosine	Y