DIMENSIONALITY REDUCTION FOR PROTEIN SECONDARY STRUCTURE PREDICTION

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER SCIENCE

By

Yasin GÖRMEZ

July 2017

SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained 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 materials and results that are not original to this work.

Name-Surname: Yasin GÖRMEZ

Signature :

REGULATORY COMPLIANCE

M. Sc. thesis titled Dimensionality Reduction for Protein Secondary Structure Prediction has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

Prepared By Advisor

Yasin GÖRMEZ Assist. Prof. Zafer AYDIN

Co-supervisor

Assoc. Prof. Oğuz KAYNAR

Head of the Electrical and Computer Enginering Program

Assoc. Prof. Vehbi Çağrı GÜNGÖR

ACCEPTANCE AND APPROVAL

M. Sc. thesis titled Dimensionality Reduction for Protein Secondary Structure Prediction and prepared by Yasin GÖRMEZ has been accepted by the jury in the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

……….. /……….. / ………..

(Thesis Defense Exam Date)

JURY:

Advisor	: Assist. Prof. Zafer AYDIN
Co-supervisor	: Assoc. Prof. Oğuz KAYNAR
Member	: Assoc. Prof. İsa YILDIRIM
Member	: Assist. Prof. Mete ÇELİK
Member	: Assist. Prof. Celal ÖZTÜRK

APPROVAL:

The acceptance of this M. Sc. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering & Science, Executive Board dated ….. /….. / ……….. and numbered …………..……. .

……….. /……….. / ………..

(Date)

Graduate School Dean

Prof. Dr. İrfan ALAN

ABSTRACT

DIMENSIONALITY REDUCTION FOR PROTEIN SECONDARY STRUCTURE PREDICTION

Yasin GÖRMEZ
MSc. thesis in Graduate School of Engineering and Science
Supervisor: Assist. Prof. Zafer AYDIN

Co-Supervisor: Assoc. Prof. Oğuz KAYNAR

July 2017

Proteins are important for our lives and they execute essential metabolic processes. The functions of the proteins can be understood by looking at the three-dimensional structures of the proteins. Because the experimental detection of tertiary structure is costly computational systems that estimate the structure provides a convenient alternative. One of the important steps of protein structure estimation is the identification of secondary structure tags. As new feature extraction methods are developed, the data sets used for this estimation can have high dimensions and some of the attributes can contain noisy data. For this reason, choosing the right number of features and the right attributes is one of the important steps to achieve a good success rate. In this study, size reduction process is applied on two different datasets using a deep autoencoder and various dimension reduction and feature selection techniques such as basic component analysis, chi-square, information gain, gain ratio, correlation-based feature selection (CFS) and the minimum redundancy maximum relevance algorithm as well as search strategies such as best first, genetic search, greedy algorithm. To evaluate the prediction accuracy, a support vector machine classifier is employed.

Keywords: Protein Secondary Structure Prediction, Autoencoder, Deep Learning, Feature Selection, Dimension Reduction

ÖZET

PROTEİN İKİNCİL YAPI TAHMİNİ İÇİN BOYUT KÜÇÜLTME

Yasin GÖRMEZ
Elektrik ve Bilgisayar Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Zafer ADIN
Eş Danışman: Doç. Dr. Oğuz KAYNAR
Temmuz 2017

Gerekli metabolik süreçleri yürüten proteinler insan hayatı için büyük önem taşımaktadır. Proteinlerin fonksiyonları ile üç boyutlu yapıları arasında yakın bir ilişki bulunmaktadır. Dört yapı düzeyi olan proteinlerin bir çoğunun, birincil yapı olarak da adlandırılan amino asit dizilimi bilinmekte ancak üçüncül yapıları bilinmemektedir. Üçüncül yapıların laboratuvar ortamında tespit edilmesinin çok maliyetli ve zor olması, amino asit dizilimini kullanarak yapı tahmini yapan sistemlerin geliştirilmesine neden olmuştur. Protein yapı tahmini yapan sistemlerin en önemli aşamalarından biri ise ikincil yapı etiketlerinin tanımlanması işlemidir. Yeni öznitelik çıkarma yaklaşımları geliştirildikçe yapısal özelliklerin tahmini için kullanılan veri setleri yüksek boyutlara sahip olabilmekte ve kullanılan özniteliklerden bazıları gürültülü veri içerebilmektedir. Bu nedenle uygun sayıda ve doğru öznitelikleri seçmek, iyi bir başarı oranı elde etmek için önemli aşamalardan biridir. Bu çalışmada iki farklı veri seti üzerinde derin oto kodlayıcı kullanılarak boyut düşürme işlemi uygulanmış, temel bileşen analizi, ki-kare, bilgi kazancı, kazanım oranı, korelasyon tabanlı öznitelik seçim teknikleri ve minimum fazlalık maksimum ilgi algoritması gibi çeşitli öznitelik seçim ve boyut düşürme teknikleri ayrıca genetik algoritma, aç gözlü algoritma ve en iyi ilk önce algoritması gibi çeşitli arama stratejileri ile birlikte kullanılarak elde edilen veri setleri ile karşılaştırılmıştır. İkincil yapı tahmin başarısının karşılaştırılması için destek vektör makinası kullanılmıştır.

Anahtar kelimeler: Protein İkincil Yapı Tahmini, Oto Kodlayıcı, Derin Öğrenme, Boyut Düşürme, Öznitelik Seçimi

Acknowledgements

I would like to thank the academicians in Abdullah Gül and Cumhuriyet University for their contribution in this thesis especially my advisor Assist. Prof. Zafer AYDIN and my co-advisor Assoc. Prof. Oğuz KAYNAR who encouraged me work in bioinformatics and always supported me during the thesis.

I want to thanks my friends especially my childhood friend Kamil ASLAN, my college friend Yusuf Şevki GÜNAYDIN, my roommate Murat Fatih TUNA, my homemates Mehmet Ali DEVECİ and Yunus Emre IŞIK, and my collegeus Ferhan DEMİRKOPARAN and Gülsüm UZUT. Lastly, I would like to thank my father Bilal, my mother Hatice, my sisters Necla, Yasemin and Merve and to Fluffy because they supported me during the whole life.

The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources)

Table of Contents

List of Figures

List of Tables

“To Our Martyr

Chapter 1

Introduction

Proteins formed by combining twenty amino acids in different sequence and different numbers are building block of humans' body. That's why they have critical importance for our lives. Proteins can also be used for making medicines and are useful in many other situations. For molecular design and biological medicine design knowing protein function is very important. In structural biology, there is a strong association between protein's function and protein's three-dimensional (3D) structure. Because determining protein structure with experimental methods is costly, prediction of the three-dimensional structure from amino acid sequence provides an effective alternative and is one of the most important aims in theoretical chemistry and bioinformatics.

Methods to predict three-dimensional structure of proteins are divided into two main categories such as; template-based modelling and free modelling. In template-based modelling, a protein, which has a specified similarity to target protein's amino acid sequence, is detected. Then, three-dimensional structure prediction is computed using that protein as a template. If such a protein cannot be specified, three-dimensional structure is predicted with free modelling. According to the thermodynamic hypothesis used by comparative and free modelling, proteins are folded to have minimum free energy in physiological environment. Several one-dimensional structural characteristics like secondary structure, profile matrix, torsion angles, and solvent accessibility are used as features to predict the 3D structure of a protein. Inferring these one-dimensional (1D) characteristics with minimum error is important for three-dimensional structure prediction. Until today many machine learning algorithms have been developed for predicting 1D properties of proteins. In machine learning dataset has critical importance for the accuracy and performance of classifier. Having too many features may increase training time and can cause overfitting, which reduces the accuracy on unseen data. Furthermore it can distort training due to noisy features. On the other hand a few features may not sufficient for satisfactory training, which is known as underfitting. Hence, proper and sufficient numbers of features have to be employed in machine learning models. To solve the aforementioned problems, dimensionality reduction techniques such as feature selection and projection methods can be used [1]. The main difference between these two techniques is that, in feature selection a subset of features are selected and used without any change, but in projection methods, the size of dataset is reduced by using all features with least information loss.

In this thesis, principal component analysis (PCA) [2], autoencoder (AE) [3], ranker chi-square (X^2) [4], ranker information gain [5], ranker gain ratio[6], minimum redundancy maximum relevance (MRMR) [7], correlation-based genetic feature selection (CFS-genetic) [8], correlation-based greedy feature selection (CFS-greedy) [9] and correlation-based best first feature selection (CFS-best first) [10] are used as dimension reduction techniques for protein secondary structure prediction (PSSP). To predict the secondary structure of proteins, a support vector machine from a two-stage classifier is employed. The organization of this thesis is as follows. Chapter 2 explains protein structure, protein structure prediction and includes literature review for protein secondary structure prediction; Chapter 3 presents methods developed in this study; Chapter 4 details the experiments and results; finally Chapter 5 provides concluding remarks and future work.

Chapter 2

Structure of Protein

Proteins form a major class of macromolecules found in every organism composed of consecutive attachment of amino acids by peptide bonds. There are twenty different amino acid types commonly found in nature. Amino acids are organic compounds that consist of a carbon atom (Ca), amine group (-NH_2), carboxyl group (COOH), and a side chain molecule (R). Figure 2.1 shows an example of an amino acid molecule. The amino acids are produced at the ribosomes and have different physical and chemical properties such as electrostatic charge they carry, the hydrophobic states, acid dissociation constants (pKa), molecular size and the functional group. These characteristics play an important role in determining the structure of proteins [11].

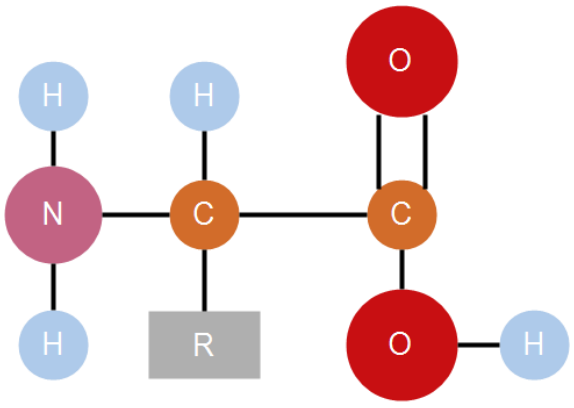

Figure 2.1 Structure of a free amino acid

2.1 Protein Structure Levels

Protein structure has four main levels: Primary structure, secondary structure, tertiary structure and quaternary structure. Primary structure is the amino acid sequence, secondary structure represents regular hydrogen bond patterns, tertiary structure is the three-dimensional structure of a single amino acid chain and quaternary structure is the three-dimensional structure of the protein, which might contains more than one amino acid chain. Figure 2.1.1 shows four levels of protein structure.

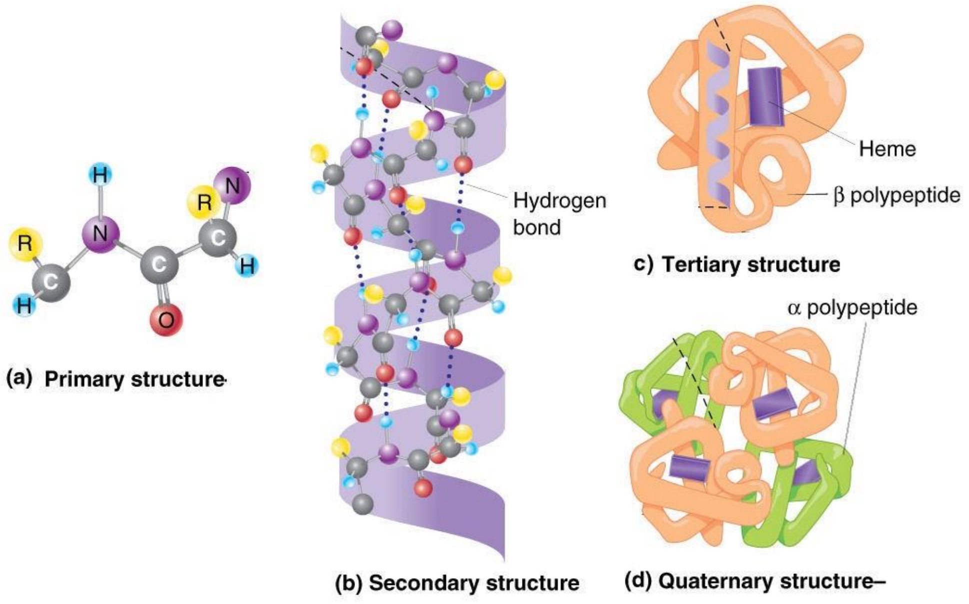

Figure 2.1.1 primary, secondary, tertiary and quaternary structures in proteins [12]

2.1.1 Primary Structure

Primary structure is the amino acid sequence of a polypeptide chain. It stays together with peptide bonds that occur during protein synthesis. The primary structure of a protein is decided in vivo by the gene that encodes its amino acid content. The amino acid sequence is serves as a signature for the protein dictating its structure and function. While this sequence can be determined by methods such as mass spectrometry (MS) or Edman degradation, typically it is identified by directly reading the sequence from the encoding gene [11].

2.1.2 Secondary Structure

Secondary structure in proteins is formed by regular hydrogen bonds between neighboring amino acids with similar dihedral angles. There are two basic motifs that form the hydrogen bond pattern such as; rotation motif and bridge motif. In the rotation motif, also referred to as the n-rotation motif, there is a hydrogen bond between an amino acid at position i and the amino acid at position $i + n$ and n typically takes values of 3, 4 or 5. In the bridge motif, there is usually hydrogen bonding between amino acids that are not closely related to each other in sequential order. Subsequent secondary structural elements are formed when the rotation and bridge motifs are successively brought to a certain layout. For example, the repeating 4-rotation motif forms the alpha helix and the repeating bridge motif forms beta strands and beta sheets. The three-dimensional structure of proteins can be thought of as the successive organization of secondary structural elements.

2.1.2.1 Helix

In this structure the protein backbone adopts a helical structure (Figure 2.1.2.1.1). There are three types of helix: Alpha helix (α-helix), 3_{10} helix and pi helix (π–helix). Helices can have various functional roles. These may include the motifs connected to DNA (strand-coil-strand, leucine zipper, zinc finger) and structures passing through the cell membrane [13].

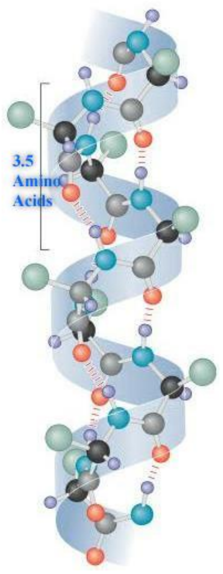

Figure 2.1.2.1.1 Alpha helix [14]

2.1.2.2 Beta Strands and Beta Sheets

Beta strands are the second most common regular units that stabilize the structure of proteins (Figure 2.1.2.2.1). A beta strand consists of a polypeptide chain that has 3 to 10 amino acids. In beta-strands, the polypeptide typically has an extended conformation. Beta strands are aligned pairwise and consecutively in three dimensional space interacting through hydrogen bonds. As a result of this interaction, beta-sheets units are formed, which contain at least two beta-strands. The interacting amino acid segments may be close to each other and linked by a short loop, or they may be separated by many other structures. Even though interacting beta strands are sequentially far from each other, they can come closer in the three-dimensional space as a result of the folding process. Protein aggregates and fibrils formed through combination of beta strands play a role in the formation of various diseases like Alzheimer's [15].

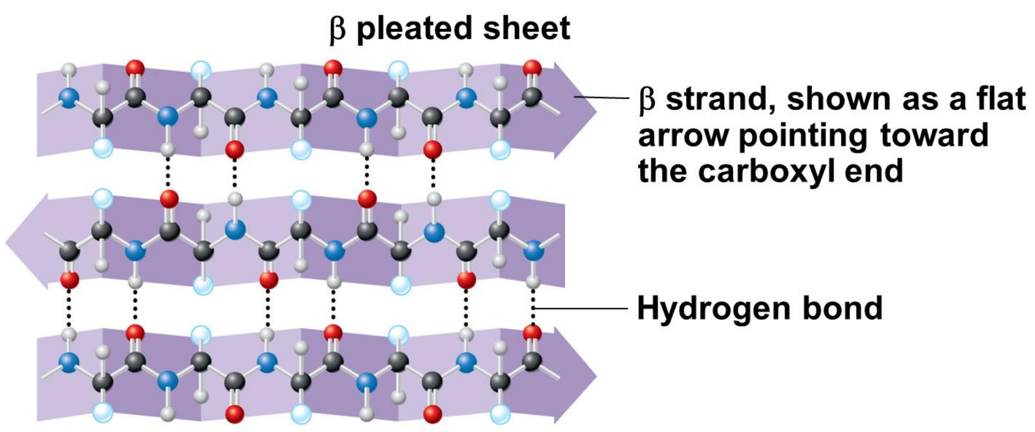
β pleated sheet
β strand, shown as a flat arrow pointing toward the carboxyl end
Hydrogen bond

Figure 2.1.2.2.1 Beta sheet [16]

2.1.2.3 Loop

Loops are structures usually located at the surface of the protein. They typically occur between helix and beta sheets with different lengths and configurations. Unlike the amino acids in the internal region of proteins, amino acids in loops are not exposed to spatial and environmental constraints. They also do not play an effective role in the regulation of secondary structural elements in the inner zone. That’s why there may be more mutations in the loops. Regions that have undergone this type of mutation in a series of alignments may indicate the loop structure. Loops are more inclined to contain cyclic charged and polarized amino acids and are usually found in the functionally active regions [17]. There are three types of loops: curl, stitch and random coil.

2.1.3 Tertiary Structure

Tertiary Structure is the 3D structure of a single protein molecule. It can be defined as the coordinates of atoms in 3D space. The strands and sheets are folded to form a compact structure. This folding is guided by hydrophobic interactions however, to stabilize the overall structure certain regions of a protein may be fixed with specific tertiary interactions [11].

2.1.4 Quaternary Structure

Quaternary structure is the agglomeration which formed by several proteins or polypeptide chain. It is stabilized by non-covalent bonds and disulfide bonds which stabilize the tertiary structure. Most proteins do not have quaternary structure and they function as a monomer. An example of a quaternary structure is the hemoglobin protein, which carries oxygen in the blood and is composed of four chains [11].

2.2 Protein Structure Prediction

Estimation of three-dimensional structure is understood when protein structure prediction is called. Since this is a rather difficult problem, instead of predicting the three-dimensional directly, the various structural properties of the protein are usually primarily estimated and then these characteristics will be used for three-dimensional structure prediction. These include the secondary structure prediction, dihedral angles prediction and solvent accessibility prediction as a general work.

2.2.1 Secondary Structure Prediction

The secondary structure prediction is the identification of the secondary structural elements starting from the sequence information of the proteins. The aim of this problem is to assign secondary structural elements (helix, beta strands, loops) for each amino acid (Figure 2.1.1). To estimate secondary structure generally supervised learning approaches are used. For this, a model is trained by using proteins that have known secondary structure. Then unknown proteins are predicted. The first developed methods of secondary structure prediction were based on the tendency of each amino acid to form helices or leaves. Sometimes, in addition, rules to estimate the formation of secondary structural elements are included. These methods were successful at 60% to predict which of the three states (helix, strands, loop) an amino acid residue would adopt. Subsequently, a significant increase in accuracy was achieved by using multiple sequence alignments and the success rate reached 80-82% [18], [19]. In addition to the multiple sequence alignment, the accuracy reached 84-85% when structural profiles were used; if there are proteins with known secondary structure elements which have similarity in various levels [20], [21]. These accuracy rates make it possible to use secondary structure estimation information in many other problems. These problems include estimating the folding class, estimating the three-dimensional structure, classifying the structural motifs, and improving the alignment of the series.

M	S	N	T	T	W	G	L	Q	R	D	I	T	P	R	L
L	L	E	H	H	E	L	H	E	L	H	L	E	L	H	E

Figure 2.1.1 Three state protein secondary structure prediction. The first line is amino acid sequence. The second line is secondary structure

2.3 Literature Review for Secondary Structure Prediction

As mentioned earlier, the secondary structure prediction is defined as assigning a secondary structure label for each amino acid of a given protein. Various machine learning methods have been developed to estimate the secondary structure including artificial neural networks (ANN), support vector machines, dynamic Bayesian networks, random forests and ensemble techniques. Salamov and Solovyev employed artificial neural network and k nearest neighbor with a scoring matrix and obtained 72.2% Q3 accuracy [22]. Jones used neural networks on position specific scoring matrices (PSSM) computed by PSI-BLAST algorithm and achieved an average Q3 score between 76.5% and 78.3% [23]. Jian-wei et al. proposed a neural network model for secondary structure prediction and compared it with traditional back propagation algorithm. As a result, they obtained %9 improvement [24]. Yaseen and Li applied a neural network on a dataset that is obtained by using statistical-context based scores and achieved 82.74% Q3 score [25]. Yao et al. obtained 78.1% Q3 accuracy by a method called DBNN that use dynamic Bayesian network and neural network [26]. Pollastri et al. employed recurrent neural network with PSI-BLAST algorithm on both three and eight classes and their model obtained 78% correct prediction [27]. Mirabello and Pollastri create application based on bidirectional recurrent neural networks called Porter 4.0 and Paleale 4.0. Porter 4.0 got 82.2% Q3 accuracy and Paleale 4.0 got 80.0% Q3 accuracy [19].

Among the other machine learning methods, k-nearest neighbor and minimum distance use a distance formula (e.g. Minkowski, Euclid etc.) to classify data. These algorithms do not require pre-training and test data are classified using training set each time. Ghosh and Parai applied k nearest neighbor, minimum distance and fuzzy k nearest neighbor algorithm on a dataset that contains amino acid sequence of protein and they compared these methods with multilayer neural networks. They showed that these methods give better accuracy than multilayer neural networks [28]. Yang at al. proposed a novel nearest neighbor method that uses non-homologous and both homologous characteristic of protein secondary structure and obtained 87.51% Q3 score [29].

Support Vector Machines (SVM) is another widely used method to estimate protein secondary structure. In SVM algorithm, which will be described in detail in the next sections of study, data is divided into two classes with the help of a linear hyper plane. Hua and Sun applied an SVM on RS126 and CB513 datasets and obtained 73.5% Q3 score [30]. Aydin et al. used SVM and Dynamic Bayesian network on CB513 and achieved 80.3% Q3 score [18]. Huang and Chen used support vector machines on a dataset that is generated using PSSM values and four physicochemical features (net charges, conformation parameters, side chain mass, and hydrophobic), then obtained 79.52% Q3 accuracy [31]. Wang et al. made parameter optimization for support vector machines for grid search and genetic algorithm. On the one hand model trained by using grid search gave 76.08% Q3 score and the model learned by using genetic algorithm produced 76.11% Q3 score [32].

Another machine learning technique used on protein secondary structure is Hidden Markov Model (HMM). It’s a technique to estimate future behavior based on current behavior. It’s widely used as a classifier in many fields such as hand-writing recognition, bioinformatics and image processing etc. Martin et al. considered finding optimal hidden Markov model for protein secondary structure prediction and obtained 75.5% Q3 accuracy [33]. Aydin et al. extended hidden semi-markov model to estimate secondary structure for single sequence and obtained 67.89% Q3 score [34].

As previously defined, protein secondary structure prediction aims to assign secondary structural elements for each amino acid. Therefore the number of samples in the datasets will be equal to the number of amino acids, which can be large. In this case, the speed of the learning algorithm becomes very important. Extreme learning machine as a very fast algorithm is a derivative of fully-connected neural network first proposed by Huang at al. [35]. Because of the speed of this algorithm, it may be used in a problem with large sample size. Wang et al. applied extreme learning machine on protein datasets CB513 and RS126 and reached 74.7% Q3 accuracy. They found that the accuracy of extreme learning machine is promising and because of the speed of algorithm, it may be used in secondary structure prediction [36].

Although the classification algorithms sometimes make similar mistakes when compared to each other, in some cases it is possible to make mistakes belonging to a specific class. In order to avoid such types of errors, ensemble methods may be used, in which two or more classification algorithms are combined by using some mathematical or statistical techniques. Lin et al. combined several support vector machines and obtained 74.98% Q3 accuracy [37]. Bouziane et al. combined artificial neural networks and support vector machines with majority voting and ideal fold selection on CB513 dataset. The model that used majority voting gave 76.58% Q3 accuracy and ideal fold selection 78.50% Q3 accuracy [38].

In the artificial neural networks, the number of neurons in the hidden layers is important for the accuracy. If there is a few numbers of neurons, the model cannot separate the samples well. To learn highly non-linear relationships the number of hidden neurons should be sufficiently high, which requires a large number of data samples in training set. This increases the computational complexity of the learning phase. To solve this problem, deep learning approaches are proposed and applied successfully in many problems. Spencer et al. used deep belief networks for protein secondary structure with 80.7% Q3 score [39]. Li and Yu developed cascade convolutional neural network on 3 different dataset. They got the best 76.9% Q8 accuracy on CB513 [40]. Wang et al. employed deep convolutional neural fields on protein dataset and they obtained 84% Q3 accuracy [41].

In machine learning, the feature set is very important. Dimension reduction and feature selection methods, which reduce the data set to a smaller size have been used in many studies to improve classification performance. Li et al. applied principal component analysis on a new dataset and obtained 86.7% Q3 accuracy by support vector machines [42]. Adamczak used t-statistics and information gain for feature selection and principal component analysis for dimension reduction. He trained a neural network with reduced data set and achieved 79.1% Q3 accuracy [43].

Researchers also applied other machine learning methods that are not widely used to estimate protein structure. Li et al. proposed a structural position-specific scoring matrix and achieved 82.7% Q3 accuracy on EVAset [21]. Zongooei and Jalili used support vector regression and support vector regression based on non-dominated sorting genetic algorithm. The first algorithm gave 85.79% and the second algorithm 84.94% Q3 accuracy on CB513 and 81.4% on an independent test data [44]. Fayech et al. proposed a technique called data mining for prediction and they obtained 78.2% Q3 accuracy [45].

Chapter 3

Methods

3.1 Classification Methods

3.1.1 Feature Extraction for One Dimensional Protein Structure Prediction

Proteins with similar amino acid sequences typically have similar structure. When the amino acid sequence is different, there is usually no structural similarity however, there are proteins, which have similar structure but their amino acid sequence is considerably different. Since the amino acid content of structurally similar proteins may be different, statistical techniques are proposed to summarize this difference. One of these is the profile matrix such as position specific scoring matrix (PSSM) that is mainly a statistical score table that, is obtained by aligning proteins in the same family, it shows which position the amino acids is seen less or frequently and contains a likelihood score for observing the 20 amino acids in each position of the query protein. In this thesis, we use PSI-BLAST PSSM, HHMAKE PSSM, and structural profile matrices to predict secondary structure of proteins.

3.1.1.1 PSI-BLAST

The PSI-BLAST method can be thought of as the iterative version of the BLAST algorithm [46]. The query proteinis aligned with the proteins in the database and the remaining proteins above the threshold are selected. In the second and subsequent iterations, the proteins above the threshold are aligned by a multiple alignment method and a statistical profile matrix is calculated and aligned with proteins in the database. In each iteration, the profile matrix is updated using proteins above the threshold. Usually 3-6 iterations are sufficient for convergence. The size of the profile matrix obtained by the PSI-BLAST method is 20*U, where U is the number of amino acids in the target (i.e. query) protein. With the use of profilers in alignment, proteins with structural similarity but without sequence similarity can be discovered and included to the profile matrix. The most widely used profile matrix derivation method for structure prediction is PSIBLAST. This can be due to the program's fast runtime, sensitivity at a certain level, easy access to the software, and regular updates on the software. Even if the PSI-BLAST method can detect more distant protein similarities, it also performs some mismatches. Therefore, profile matrices produced by using this method contain noise. The methods and databases are open for access and will be downloaded from the relevant internet address [47].

3.1.1.2 Profiles based on Hidden Markov Model

Profiles derived from hidden Markov models (HMM) can also be used as input features for predicting structural properties of proteins [48]. Profiles based on hidden Markov models are known to be more sensitive than standard profiles and are able to discover more distant protein relations. In this thesis, hidden Markov models obtained from the first iteration of HHBlits method is transformed into position specific scoring matrices (PSSM) which, dimension of 20*U and used as the second profile matrix. To generate HHMAKE profiles (position specific scroring matrix), in the first step, proteins are aligned against the NR database by the HHBlits algorithm and proteins above the threshold are multiply aligned. Then, an HMM-profile model is obtained from this multiple alignment and the distributions in match states are normalized to interval [0, 1] to derive the HHMAKE profile. The HHblits method and databases can be downloaded from the internet address of this software [49].

3.1.1.3 Structural Profiles

In addition to profile matrices based on multiple alignments of amino acid sequences, structural profile matrices can also be used for 1D structure prediction [50]. Structural profile matrices are constructed using structural sequences of proteins found by sequence alignment methods. Typically, the dimension of a structural profile matrix constructed for secondary structure estimation is 3 x U (U is the number of amino acids in the target protein and each column has the observation score of one of the three secondary structures for that amino acid). Structural profiles can be evaluated in separate categories from methods that use only sequence profiling, since they also use structural information of template proteins that are similar to the target protein. In another category, structure labels of target protein can be predicted by using secondary structure information of template proteins, for which the sequence similarity with the target is below a certain level. In this case, the use of structural profiles is in between these two categories.

The accuracy of prediction is directly related to the similarity level between protein sequences used in constructing the structural profile matrix and the target protein. This similarity can also be local in which target resembles a sub-region of a database protein. On this basis, structural information of local similarities can be used to estimate the structural properties of the target protein. In this thesis, the HMM-profile model obtained for each target is aligned with the HMM-profiles of the PDB proteins, which is achieved by the second phase of the HHBlits method. In the next step, those templates for which the percentage of sequence identity score is above 20% are eliminated. Then the structural profile matrix is derived by computing the average weighted frequency of secondary structure labels aligned to each amino acid of the target. Finally the profile matrix is normalized so that each column sums to 1 [51].

3.1.2 DSPRED Method

The DSPRED method is a two-stage classifier that includes Dynamic Bayesian Networks (DBN) and a support vector machine classifier.A separate DBN is trained for each position specific scoring matrix produced by PSI-BLAST [47] and the first step of HHBlits [48] (PSIBLAST PSSM and HHMAKE PSSM). Then the predictions are combined with a structural profile matrix obtained from the second stage of the HHBlits method and sent as input to an SVM classifier. Figure 3.1.2.1 summarizes the steps of DSPRED.

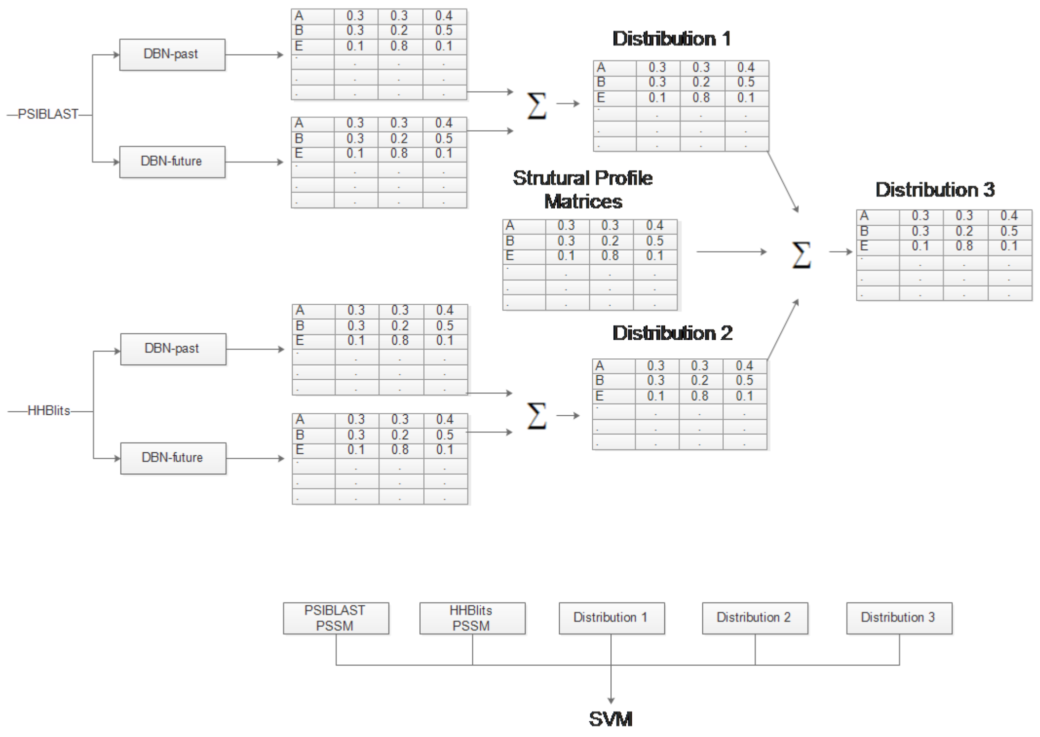

Figure 3.1.2.1 Steps of DSPRED method for 1D protein structure prediction

In this figure, DBN-past represents the model, in which the profile vector in current position depends on the neighboring positions that come before, and DBN-future represents the model, in which the profile vector in current position depends on the positions that come after. Here the vectors are the columns of the profile matrix and there are as many columns as the number of amino acids. As a result, two types of DBN models are trained for each profile matrix (totally four). Then, the probability distributions for the secondary structure classes from these DBN classifiers are averaged over various combinations. For example, the average of the estimated distributions from the PSIBLAST profile matrices is Distribution 1, Dynamic Bayesian Network estimates by using HHMAKE profile matrix that is produced in the first phase of HHBlits is Distribution 2, and the average of Distribution 1, Distribution 2, and structural profile matrices is Distribution 3. Structural profile matrix is produced in the second phase of HHBlits by aligning the HMM-profile matrix from the first phase to the HMM-profile matrices of proteins in PDB [52] and normalizing the frequencies of the tag information of PDB proteins. In this thesis, dimension of distribution 1, 2, and 3 and structural profile matrix is 3*U because we make three state protein secondary structure prediction. Therefore, each column contains the estimated probabilities of the secondary structure classes at that position. In the second stage of DSPRED, the profile matrices (PSI-BLAST and HHMAKE) used for DBN are combined with Distributions 1, 2, and 3 and sent to a support vector machine. For this purpose, a sliding and symmetric window around each amino acid is selected and the columns of the profile matrices, distribution 1, 2, and 3 corresponding to these windowed positions are used as input parameters. Finally, the support vector machine predicts the secondary structure class of the amino acid in the center of the window.

3.1.3 Support Vector Machines

Support Vector Machine (SVM) is a learning method proposed by Vapnik for the solution of classification and curve fitting problems, which takes advantage of statistical learning theory and the principle of minimizing the structural risk [53]. This method is commonly used for determining classes that are linearly separable but can also be used for nonlinear classification thanks to the kernel functions that map the input space to a higher-dimensional. Support vector machine are supervised machine learning methods aim to separate data in two classes by using a linear hyper plane. In the learning phase the parameters of these separators are determined then unknown classes are estimated using these parameters.

The main purpose of the SVM is to determine the best separator that has the minimum error. As shown in figure 3.1.3.1, two support vectors that minimize the error are selected and the distance between the planes that pass through the closest support vectors is maximized. In the last step, the class value for new sample is calculated by using equation 3.1.3.1. In this equation y represents the class value, x represents the new sample's feature vector, w represents the weight vector that is perpendicular to the hyperplane and b represents the constant value. If y>0 the sample is assigned to the first class, otherwise it is assigned to the second class.

$$y = w^t x + b \qquad \text{(eq. 3.1.3.1)}$$

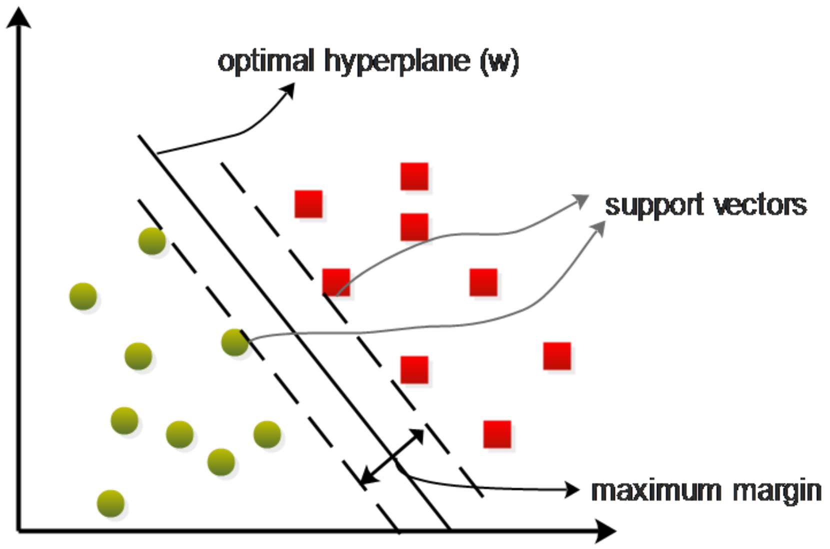

Figure 3.1.3.1 Support vector machines and hyper plane selection

Unlike other machine learning techniques, support vector machines can separate only two classes. For three or more classes, two techniques can be used: one versus all (OVA), or one versus one (OVO). In OVA, one class is selected as the first class and all the remaining classes as the second class. A separate model is trained for each class and predictions from individual classifiers are combined. In OVO, separate SVM is trained for each class pair and predictions from individual classifiers are combined.

3.2 Feature Selection Techniques

3.2.1 Chi Square (X^2)

The chi-square method, also known as X^2 test, is developed by Randy Kerber in 1992 and also by Huan Lui and Rudy Setiono in 1995. It can be used to determine whether variables are eligible to represent the dataset or not [4]. In a chi-square test there are two hypotheses, H_0 and H_1. H_0 represents that variables are not eligible (i.e. null model), and H_1 represents that variables not eligible to represent the dataset. The chi-square method has two phases. In the first step, the chi-square statistic of the observed values with respect to the actual classes is calculated. X^2 value can take values between zero and positive infinity. If this value is close to zero, the observed frequency values and the expected frequency values are comparably close. If this value is high, the observed frequency values and the expected frequency values differ significantly. For this reason, in the second stage, the X^2 value is compared with a threshold value determined from the Chi-square distribution. This threshold value is determined based on the level of significance and the degree of freedom. The significance level represents the probability of obtaining a chi-square statistic greater than the threshold using the null model and the degree of freedom is calculated by subtracting one from the number of attributes being analyzed. The H_1 hypothesis is accepted if the calculated value is greater than the specified value. Otherwise H_0 is accepted. Equation 3.2.1.1 shows the formula of the chi square statistic.

$$X^2 = \sum_{i=1}^{i=n} \frac{(o_i - e_i)^2}{e_i} \qquad \text{(eq. 3.2.1.1)}$$

In this equation, n represents the number of features in dataset; o_i represents the observed frequency value for i^{th} feature, and e_i represents the expected frequency value for i^{th} feature.

3.2.2 Information Gain

The information gain (IG) is one of the entropy-based metrics that can be used to calculate the estimated loss when the data set is divided by attributes [5]. Entropy is a value that determines the irregularity or uncertainty of the system. A high entropy value indicates that the system contains high information. To compute the information gain metric, the entropy value for class labels of a given data set is computed as formulated in equation 3.2.2.1.

$$E = -\sum_{i=1}^{n}\left(\log_2 \frac{fs(i)}{N}\right) \times \frac{f(i)}{N} \qquad \text{(eq. 3.2.2.1)}$$

In this equation, n represents the number of class in dataset, $fs(i)$ represent the number of sample for i^{th} class, and N represents the total number of samples. In the second phase of calculating information gain metric, the entropy value for each attribute is calculated and this new entropy value is subtracted from the value found in the first step to calculate the information gain for each attribute. Equation 3.2.2.2 shows the calculation of entropy value for each attribute, and equation 3.2.2.3 shows calculation of information gain value.

$$E(i) \sum_{k=1}^{n} \frac{ss(k)}{N} \times \sum_{m=1}^{cs} -\frac{csk(k,m)}{ss(k)} \left(\log_2 \frac{csk(k,m)}{ss(i)}\right) \qquad \text{(eq. 3.2.2.2)}$$

$$B(i) = E(i) - E \qquad \text{(eq. 3.2.2.3)}$$

In the equations, $E(i)$ represents the entropy value for i^{th} feature, n represents the number of unique value of the i^{th} feature, $ss(k)$ represents the number of sample that belongs k^{th} value of i^{th} feature, N represents the total number of sample in dataset, cs represents the number of class in dataset, $csk(k,m)$ represents the number of sample that belongs the feature i, variable k and class m, $B(i)$ represents the information gain for i^{th} feature, and E represents the value calculated in eq. 3.2.2.1.

3.2.3 Gain Ratio

The gain ratio (GR) is also one of the entropy-based metrics that can be used to calculate the estimated loss when the data set is divided by attributes. When an attribute in the dataset has many different values, the number of samples falling for each value is low for that attribute. For this reason, the entropy value calculated for that attribute becomes small and the information gain large. As explained in the information gain method, the large value of this variable indicates that the variable is good at defining the dataset. If there are a lot different values for an attribute, information gain methods select that attribute as a good separator. Although the system memorizes the training set well, it cannot separate the test set properly using that attribute. As a solution to this problem, the gain ratio normalizes the information gain with the partitioning information for each attribute. If the gain ratio of an attribute is high, then this attribute is a good separator. Equation 3.2.3.1 formulates how partitioning information value is calculated and equation 3.2.3.2 shows the computation of gain ratio.

$$S(i) = -\sum_{k=1}^{n} \frac{ss(k)}{N} \times (\log_2 \frac{ss(k)}{N}) \qquad \text{(eq. 3.2.3.1)}$$

$$K(i) = \frac{B(i)}{S(i)} \qquad \text{(eq. 3.2.3.2)}$$

In the above equations, $S(i)$ represents the partitioning information value for i^{th} feature, n represents the unique values for i^{th} feature, $ss(k)$ represents the sample size that belongs the k^{th} value of i^{th} feature, N represents the total number of samples, $K(i)$ represents the gain ratio value for i^{th} feature, and $B(i)$ represents the information gain value that calculated in eq. 3.2.2.3.

3.2.4 Minimum Redundancy Maximum Relevance

Minimum redundancy maximum relevance, which proposed by Ding and Peng, is a feature selection algorithm that aims to eliminate redundant attributes and selects the attribute that is the most related with the class labels [7]. In other words it selects the attributes that have minimum correlation with each other. In the first step of the algorithm, for each x, y (two variables in the dataset) the mutual information value (I) is calculated as shown in equation 3.2.4.1.

$$I(X, Y) = \sum_{i=1}^{n} \sum_{j=1}^{n} p(x_i, y_j) \times \log \frac{p(x_i, y_j)}{p(x_i) \times p(y_j)} \qquad \text{(eq. 3.2.4.1)}$$

In this equation, n represents the number of samples in dataset, $p(x_i, y_j)$ represents the dependent probability distribution value, $p(x_i)$ and $p(y_j)$ represent marginal probabilities for the relevant sample. In this algorithm, two conditions must be provided by using I value: Minimum redundancy (mRed), and maximum relevance (mRel). mRed and mRel values are calculated as shown in equations 3.2.4.2 and 3.2.4.3, respectively.

$$mRed = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} I(x_i, y_j) \qquad \text{(eq. 3.2.4.2)}$$

$$mRel = \frac{1}{m} \sum_{i=1}^{m} I(x_i, h_i) \qquad \text{(eq. 3.2.4.3)}$$

In these equations, m represents the dimension of the new feature set, and h_i represents the class label for sample i. Finally, the features are selected to satisfy two conditions, $\max(mRel - mRed)$ and $\max\left(\frac{mRel}{mRed}\right)$.

3.2.5 Search Methods

3.2.5.1 Genetic Algorithm

The genetic algorithm (GA) is a global search optimization technique that is inspired by natural selection, crossover, and mutation events in biology. In this method, firstly, a population, which is formed by candidate solutions, is produced and this population is updated through genetic processes called selection, crossover, and mutation until the specified stopping criterion is met. Genetic algorithm uses the idea of surviving the best while finding better solutions. Genetic algorithm is different from conventional non-linear optimization techniques in that it searches for a solution population by updating it, instead of gradually changing a single solution. Because conventional optimization algorithms deal with the local properties of the iteration points, they can easily be fitted to local extremum points. Conversely, genetic algorithm uses the random search operator in addition to the systematic search, therefore it is prevented from being attached to the local minimum or maximum point.

Genetic algorithm starts with a series of solutions to optimize the parameters. Each parameter of the chromosomes that forms the solution is called the gene. Parameters can be encoded as a binary bit string, integer or real number. Without any prior knowledge, each chromosome in the first population is randomly generated using uniform distribution. Then fitness value for each solution is calculated by the determined function and solutions are sorted with respect to their fitness value. With the help of these sequential generations, new generations are produced using techniques such as mutation and crossover and these processes are repeated on new generations until the desired success rate is achieved. Because of these characteristics, a genetic algorithm may achieve a higher success rate even though it works slower than many algorithms.

3.2.5.2 Greedy Algorithm

The aim of the algorithms that use the greedy approach is to choose the best component to reach the result. This approach, which is used in many problems as graph theory, can be used as a feature selection algorithm, which can be used in two ways such as; forward feature selection (FFS), and backward feature selection (BFS).

Forward feature selection starts with the empty feature set and features are added to this set in each step, hence the name. In the first step of this algorithm the features are sorted based on a condition, which can be leave-one-out cross validation accuracy for each feature. Then, the algorithm adds the feature that has the best condition score to the empty set and calculates the condition score again. In the second step, the next feature from sorted feature set is added to the feature set and condition score is calculated again. If the new condition score is better than the previous one, this feature stays in the feature set, otherwise, it is not added. The second step is applied for each feature in the dataset and as a result a feature set which has less or equal number of dimensions than the original feature set is obtained. The aim of the backward feature selection is the same as that of the forward feature selection and its running phase very similar to forward feature selection but it works in reverse order. In the first step features are sorted as in the forward feature selection. Then the condition score is calculated for the full feature set. The feature that has the worst condition score is removed and the second condition score is calculated. If it is better than the first one, this feature is deleted from the feature set, otherwise, it is retained. These operations are applied iteratively for each feature and the final feature set is obtained.

3.2.5.3 Best First Feature Selection

Best first algorithm, which is proposed by Xu et al. as a feature selection algorithm, is very similar to greedy algorithm [10]. Unlike greedy algorithm, the best first algorithm can start from any point and search both in forward and backward directions (by considering all possible single attribute additions and deletions at a given point) [54]. Direction and starting feature set is defined as a parameter in this algorithm, which is summarized in figure 3.2.5.3.1 [55].

```
$P = \emptyset$
$best = null$
Randomly pick a node $v$
Train $\mathcal{A}$ on $\mathcal{T}$ using $v$ to maximize $\Lambda(\mathcal{T}, \mathcal{A}_v)$
Add $v$ to $P$
while $|P| > 0$
    $v \leftarrow \arg\max_{u \in P} \Lambda(\mathcal{T}, \mathcal{A}_u)$
    Remove $v$ from $P$
    if $\Lambda(\mathcal{T}, \mathcal{A}_v) > \Lambda(\mathcal{T}, \mathcal{A}_{best})$ then $best \leftarrow v$
    if $best$ did not change in the last $R$ rounds
      then STOP and RETURN $best$
    for each of $v$'s neighbors $u$
      Train $\mathcal{A}$ on $\mathcal{T}$ using $u$ to maximize $\Lambda(\mathcal{T}, \mathcal{A}_u)$
      add $u$ to $P$
    end for
end while
RETURN $best$
```

Figure 3.2.5.3.1 Best first search algorithm

In this figure, *A* is the classifier, *T* is the sample in dataset, *Λ* is the any ranking algorithm, and *P* is the selected feature set. The aim of the algorithm is to return best feature set that satisfy the condition.

3.3 Projection Techniques

3.3.1 Principal Component Analysis

Principal component analysis (PCA) is a dimension reduction technique used to find the dependency between variables. The aim of this method is to minimize the loss while maximizing variance. In the first phase, the covariance between each variable pair is calculated as formulated in equation 3.3.1.1. The covariance represents the mutual exchange of two variables. If this value is positive, the two variables grow or shrink together. If it is negative, one of the variables grows while the other reduces. If zero, they are independent from each other [2].

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X}) \times (Y_i - \bar{Y})}{n-1} \qquad \text{(eq. 3.3.1.1)}$$

In this equation, X and Y represent two variables (i.e. features), n represents the sample size in dataset, X_i and Y_i represent the value of the i^{th} sample for X and Y, respectively, and $\bar{X}$ and $\bar{Y}$ represent the mean value of X and Y, respectively. Then, the covariance matrix shown in figure 3.3.1.1 is constructed using the covariance values found in step one.

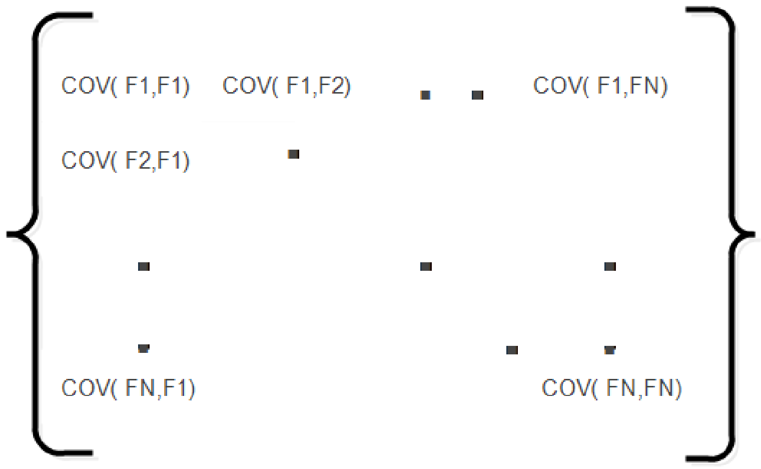

Figure 3.3.1.1 Covariance matrix

In the third step, eigenvalues and eigenvectors of this matrix are calculated and are sorted from highest to smallest to obtain the component matrix. The purpose of this sorting is to rank variables according to the representation capacity of the dataset. In the last step, the components having the highest representative value are selected. In this step number of components can be determined any value between 1 to number of feature in dataset. By multiplying this selected matrix with the feature matrix, the new feature vector that has lower dimension is obtained.

3.3.2 Deep Autoencoders

The autocoder (AE), a derivative of artificial neural networks, was first proposed by the Hinton and PDB groups in the 1990s [3]. In 2016, it became one of the main topics in machine learning when the deep learning architecture became more popular [56]. The autocoder is a fully connected artificial neural network consisting of three layers: input layer, the hidden layer and the output layer. The number of neurons in the input layer and output layer are the same and equal to the number of features in the dataset. The number of neurons in the hidden layer can be determined as desired, which is an important factor affecting the performance of the network. An example autoencoder architecture with 3 neurons in the hidden layer for a data set with 5 attributes is shown in figure 3.3.2.1.

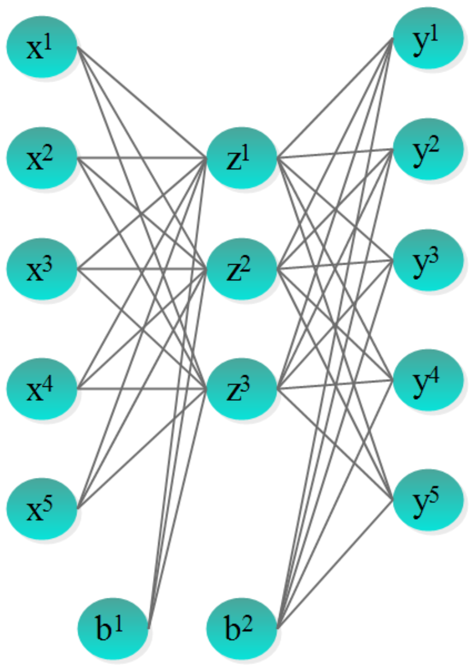

Figure 3.3.2.1 Autoencoder architecture

Autoencoder does not need any class label because it uses the input data as the output data. That’s why it is an unsupervised learning method. The network determines the optimal weight values using the backpropagation algorithm during training to match the input data to the same data at the output. For this reason, the method is also referred to as the backpropagation algorithm without a teacher [57]. The autocoder operates as a coder that maps its input data to itself with minimal loss. If there are fewer neurons in the middle layer than the output and input layer, the reduced data is derived from the middle layer. The forward propagation from one layer to the next is formulated in equation 3.3.2.1.

$$y_j = f\left(\sum_{i=1}^{n} x_i \times w_{ij}\right) + b \qquad \text{(eq. 3.3.2.1)}$$

In equation 3.3.2.1, x_i represents the value of i^{th} neuron in the current layer, y_j represents the value of j^{th} neuron in the next layer, w_{ij} represents the weights value that connect the x_i to y_j, n represents the number of neurons in the current layer, b represents the bias value (that is constant for each layer), and f represents the activation function (gauss, sigmoid, softmax etc.).

$$\min\left(\sum_{i=1}^{n}(y'_j - y_j)^2\right) \qquad \text{(eq. 3.3.2.2)}$$

During model training the weights are updated to minimize difference between the actual values and the output values expressed in equation 3.3.2.2. In equation 3.3.2.2, y'_j represents the actual value and y_j represents the value that is produced by the network.

The deep autoencoder (deepAE) is obtained by connecting several auto encoders one after another. As shown in figure 3.3.2.2, the values obtained from the hidden layer of the first autoencoder model are connected to the input layer of the second autoencoder. In deep auto encoders, each autoencoder model is trained one after another. Standard autoencoder reduces the data in one step. Hence either the dimension reduces suddenly or the reduction is little. In this case, the deep auto-encoders can be used to reduce data to lower dimensions gradually, which enables more complex datasets to be separated. This is the most important advantage of deep auto encoders.

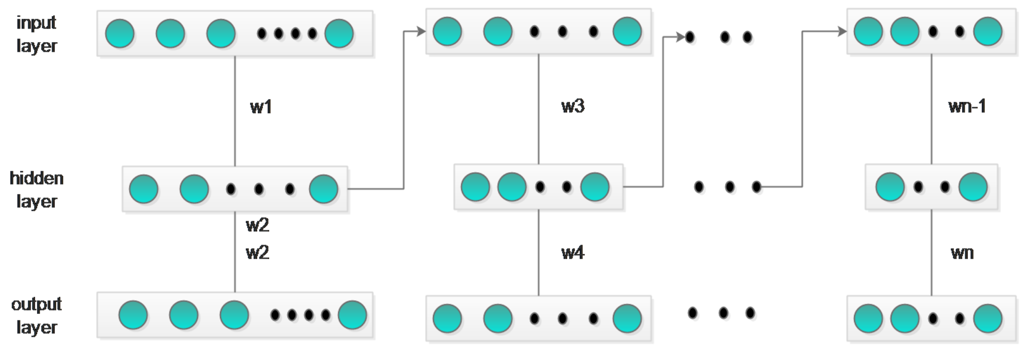

Figure 3.3.2.2 Deep autoencoder architecture

In each autoencoder model, weights that connect neurons in input layer to neurons in hidden layer are called the encoder weights, and weights that connect neurons in hidden layer to neurons in output layer are called the decoder weights. After training, data that has reduced dimension can be obtained using encoder weights. The aim of the decoder weights is to reproduce data in the original dimension. However new data produced at the output layer may not be exactly the same as the original data. In deep autoencoder model, training of each autoencoder model independently may cause a decrease in the success rate. This problem can be solved by a method called fine tuning. There are two approaches for fine tuning. In the first one, input layer of the first autoencoder model is connected to the input layer of the second autoencoder by using the encoder weights of the first autoencoder. Then for the second and third autoencoders, these steps are repeated and new network that has m layers (m represents the number of autoencoders) is obtained by applying these steps to each auto-encoder model. After this, the input layer of the last autoencoder is connected to the hidden layer of last autoencoder by using encoder weights of the last autoencoder. Then, output layer with n neurons (n represents the number of unique class labels) is generated and is connected to the new network with random weights (Shown in figure 3.3.2.3). In the final step this new network is trained by using dataset that has the class labels. In this way, deep autoencoder also becomes dependent with class label. This approach generates neural networks with initial weights, which can also be used to initiate weights for escaping local minimum in neural networks structure [58].

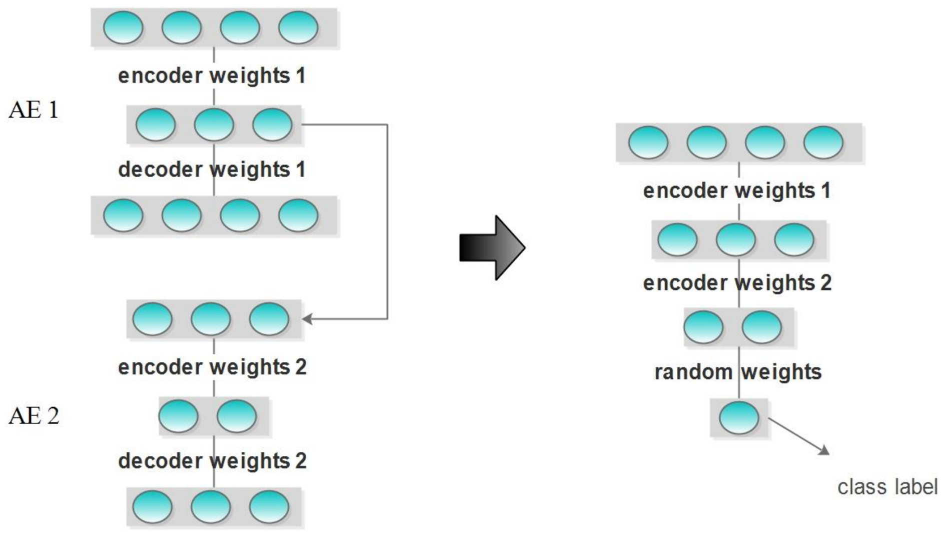

Figure 3.3.2.3 Transefer of deep autoencoder to neural network with class label for finetuning

To perform finetuning using the first approach, an additional classification layer (e.g. softmax) is needed, so the model becomes supervised. However in this approach the decoder weights cannot be updated. The second approach can be used to do finetuning without making the model supervised by updating decoder weights. Firstly, the input layer of the first autoencoder is connected to the input layer of the second autoencoder by using encoder weights of the first autoencoder. Then for the second and third autoencoders, these steps are repeated and new network that has m layers (m represents the number of autoencoder) is obtained. Finally, the input layer of the last autoencoder is connected to the hidden layer of last autoencoder by using encoder weights of the last autoencoder. These steps are similar to the first approach but, instead of a defining an output layer for class label, the last layer of the new network (it is the same as the hidden layer of the last autoencoder) is connected to the output layer of the last autoencoder by using decoder weights of the last autoencoder, then output layer of the last autoencoder is connected to the output layer of penultimate autoencoder by using decoder weights of the penultimate layer. These steps are repeated progressively till the first layer and a network with $2 \times m + 1$ layers is obtained (figure 3.3.2.4). Finally this network is trained by using the dataset without labels.

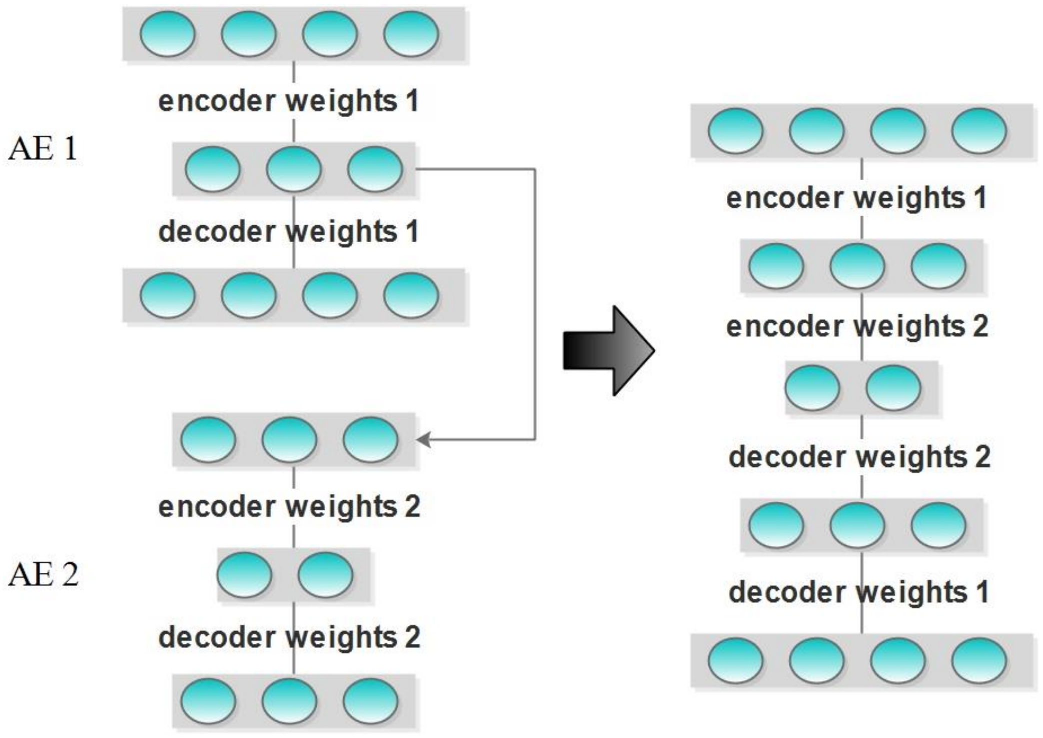

Figure 3.3.2.4 Transefer of deep autoencoder to neural network without class label for finetuning

In these type of autoencoders, the goal is to in the output layer, so that the training set may be memorized by the network. In some cases, this approach may not have high accuracy rate on the test data. To solve this problem, denoising autoencoder is used, which is a stochastic version of the autoencoder [59]. In order to enforce the hidden layer to discover more robust features and prevent it from simply learning the identity, the autoencoder is trained to reconstruct the input from a corrupted version of it. There are two main steps in the denoising autoencoder: encode the input data and decode the encoded data to the noisy version of the input data. To generate noisy data there are many methods such as adding Gaussian noise to input data. Vincent et al. randomly select some sample and they also randomly select some features from the selected sample. Then, they set the value of those features to the zero to make denoising autoencoder [60]. The system can also learn data with different patterns that are not in training data thanks to the autoencoder. In this thesis we transfer of deep autoencoder to neural network without class label forfinetuning.

Chapter 4

Experiments and Analysis

In this thesis, deep autoencoder is used as a dimensionality reduction technique for protein secondary structure prediction and is compared with the traditional feature selection and dimension reduction techniques. As well as with the model that is trained with the original dataset. As the protein dataset CB513 produced by Cuff and Barton [61] and Evaset [62] are used. For all train-test models, a one-versus-one support vector machine is used as the classifier. Q3 accuracy, precision, and recall [63] is used as the performance measures.

In this thesis cross validation [64] is used to evaluate the prediction accuracy. A 7-fold cross-validation experiment is performed on CB513 and a 10-fold cross-validation on EVAset. Proteins are randomly assigned to train and test sets for each fold. Then, from each train set 10% of the proteins are chosen randomly to form validation set and the rest is saved as the train set for optimization (i.e. model optimization or to optimize the number of dimensions). This train set contains approximately 90% of the proteins in the original train set of the cross-validation. To further reduce the sample size and speed up the optimizations, each train set for optimization is further reduced by selecting 25% of the proteins randomly. Similarly, 50% of proteins are selected randomly from each validation set. As a result, 4 different dataset created for each fold (totally $7\times4+10\times4 = 68$): such as, train set, test set, train set for optimization set and validation set for optimization.

True secondary structure labels of proteins in CB513 and EVAset are computed by the DSSP program [65] starting from 3D coordinate information in PDB.

Then for each protein PSSM and structural profiles are extracted using PSI-BLAST and HHBlits as explained in Section 3.1.1. In the next step, the secondary structure is predicted using the first phase of DSPRED method and a total of three distributions are obtained as described in Section 3.1.2. As a result, three distributions with length L ×3 and two PSSM matrixes in length L ×20 are obtained for each dataset (L representsthe number of amino acids in target protein). To extract feature for each amino acid a symmetric window of size 11 is chosen around each amino acid for CB513 dataset and a window of size 19 for Evaset.

That means for each amino acid in CB513 there are 20×11=220 HHMAKE PSSM values, 20×11=220 PSIBLAST PSSM values, 3×11=33 predicted distribution by DBNs using PSIBLAST PSSMs (distribution 1), 3×11=33 predicted distribution by DBNs using HHMAKE PSSMs (distribution 2) and 3 ×11=33 average of predicted distributions and structural profile matrix (distribution 3). Totally there are 539 features for CB513. For each amino acid in EVAset there are 20×19=380 HHMAKE PSSM values, 20×19=380 PSIBLAST PSSM values, 3×19=57 predicted distribution by DBNs using PSIBLAST PSSMs (distribution 1), 3×19=57 predicted distribution by DBNs using HHMAKE PSSMs (distribution 2), and 3×19=57 average of predicted distributions and structural profile matrix (distribution 3). Totally there are 931 features for EVAset. And in CB513 there are 84119 amino acid samples and in Evaset there are 584595 amino acid samples.

In the third phase, for each each fold a one-vs-one SVM is trained in original dimension (539 features for CB513, 931 features for Evaset) on train sets. Gamma parameter is set to 0.00781 for each fold and C parameter is set to 1, which were optimized by Aydin et al. for CB513 [6]. Then predictions are computed on test sets. The accuracy values for 7-fold cross-validation experiment on CB513 is shown in Table 4.1 and the accuracy values for 10-fold cross-validation on EVAset is summarized on Table 4.2.

	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	0.837	0.671	0.860	0.890	0.838	0.732	0.804
Fold-2	0.818	0.711	0.850	0.869	0.836	0.754	0.805
Fold-3	0.824	0.698	0.871	0.896	0.848	0.748	0.814
Fold-4	0.828	0.710	0.853	0.880	0.823	0.763	0.812
Fold-5	0.812	0.741	0.826	0.880	0.780	0.761	0.802
Fold-6	0.821	0.720	0.853	0.891	0.808	0.767	0.814
Fold-7	0.853	0.719	0.855	0.903	0.841	0.760	0.828
Mean Result	0.829	0.710	0.852	0.888	0.824	0.756	0.812

Table 4.1 Accuracy measures in original dimension evaluated by 10-fold cross validation experiment on CB513

	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	0.865	0.772	0.840	0.880	0.840	0.796	0.833
Fold-2	0.862	0.796	0.847	0.898	0.845	0.795	0.841
Fold-3	0.876	0.785	0.839	0.899	0.832	0.797	0.843
Fold-4	0.861	0.779	0.854	0.896	0.852	0.792	0.841
Fold-5	0.861	0.780	0.843	0.894	0.836	0.793	0.837
Fold-6	0.862	0.774	0.848	0.900	0.836	0.789	0.837
Fold-7	0.859	0.779	0.855	0.892	0.842	0.800	0.839
Fold-8	0.869	0.777	0.827	0.888	0.809	0.798	0.834
Fold-9	0.870	0.793	0.839	0.891	0.833	0.804	0.839
Fold-10	0.859	0.760	0.854	0.899	0.840	0.787	0.836
Mean Result	0.865	0.780	0.845	0.894	0.837	0.795	0.838

Table 4.2 Accuracy measures in original dimension evaluated by 10-fold cross validation experiment on EVAset

In the fourth phase, ranker feature selection techniques (Chi-square, Information Gain, and Gain Ratio) are applied on the each train set for optimization separately and features are sorted according to the calculated rank values by using train set for optimization. Then, features are selected by using all feature selection methods. In this phase, CFS used with search techniques such as: genetic algorithm, greedy algorithm, best first algorithm, and minimum redundancy maximum relevance are applied on the train set for optimization to find best feature set followed by selecting the same features in the corresponding validation set. As a test data, validation set is used if necessary. For each ranker method, a wrapper approach is applied, in which an SVM is trained with one-dimensional train set for optimization that has the feature with the best rank value only, and tested on the validation set that has the same feature only. Then, other features added one by one in to the datasets according to the rank order, and train and test steps are repeated for these new sets. Finally, the feature set that gives the best prediction accuracy on validation set is found. In minimum redundancy maximum relevance algorithm firstly features are ranked by using MRMR metric then best features selected with forward feature selection techniques by using correlation based feature selection algorithm. The feature selection steps described above are repeated for each fold of the cross-validation experiment. We used following command lines for feature selection algorithms [66].

Chi-Square: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.ChiSquaredAttributeEval" –S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1"*

Information gain: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.InfoGainAttributeEval" –S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1"*

Gain ratio: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.GainRatioAttributeEval" –S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1"*

Genetic algorithm: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.CfsSubsetEval -M" -S "weka.attributeSelection.GeneticSearch -Z 20 -G 20 -C 0.6 -M 0.033 -R 20 -S 1"*

Greedy algorithm: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.CfsSubsetEval -M" -S "weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1*"

CFS and best first strategy: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.CfsSubsetEval -M" -S "weka.attributeSelection.BestFirst -D 1 -N 5"*

Minimum Redundancy Maximum Relevance Algorithm: *weka.filters.supervised.attribute.AttributeSelection -b -i $train_set_file -o $train_out_file -r $test_set_file -s $test_out_file -E "weka.attributeSelection.CfsSubsetEval -M" -S "weka.attributeSelection.RerankingSearch -method 2 -blocksize 20 rankingMeasure 0 -search \"weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1 -num-slots 1\""*

After the attributes are selected, for each fold of the cross-validation experiment an SVM is trained on the original train set and class labels of test set are predicted. The experiment results on test data are shown in Tables 4.3 – 4.16.

	Dimension	**Recall ‘L’**	**Recall ‘H’**	**Recall ‘E’**	**Precision ‘L’**	**Precision ‘H’**	**Precision ‘E’**	**Accuracy**
Fold-1	81	0.836	0.676	0.860	0.888	0.836	0.737	0.805
Fold-2	91	0.822	0.727	0.844	0.865	0.830	0.763	0.808
Fold-3	78	0.822	0.724	0.861	0.890	0.843	0.755	0.816
Fold-4	531	0.828	0.711	0.853	0.880	0.823	0.763	0.812
Fold-5	404	0.815	0.743	0.830	0.884	0.783	0.764	0.806
Fold-6	13	0.825	0.735	0.834	0.870	0.797	0.775	0.810
Fold-7	485	0.853	0.723	0.857	0.905	0.842	0.762	0.830
Mean Result	----	0.830	0.720	0.848	0.884	0.821	0.760	0.813

Table 4.3 Accuracy measures of chi-square method evaluated by 7-fold cross validation experiment on CB513

	Dimension	**Recall ‘L’**	**Recall ‘H’**	**Recall ‘E’**	**Precision ‘L’**	**Precision ‘H’**	**Precision ‘E’**	**Accuracy**
Fold-1	149	0.851	0.752	0.840	0.876	0.836	0.782	0.824
Fold-2	121	0.851	0.779	0.847	0.896	0.840	0.780	0.833
Fold-3	130	0.862	0.766	0.840	0.898	0.826	0.782	0.835
Fold-4	184	0.852	0.764	0.850	0.891	0.846	0.781	0.833
Fold-5	309	0.862	0.773	0.846	0.895	0.838	0.790	0.836
Fold-6	421	0.860	0.770	0.850	0.900	0.837	0.786	0.836
Fold-7	185	0.848	0.765	0.855	0.889	0.840	0.789	0.832
Fold-8	539	0.870	0.774	0.829	0.888	0.810	0.797	0.834
Fold-9	115	0.861	0.775	0.840	0.889	0.830	0.791	0.832
Fold-10	286	0.856	0.751	0.857	0.897	0.844	0.782	0.834
Mean Result	----	0.858	0.767	0.846	0.892	0.835	0.786	0.833

Table 4.4 Accuracy measures of chi-square method evaluated by 7-fold cross validation experiment on EVAset

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	81	0.836	0.676	0.860	0.888	0.836	0.737	0.805
Fold-2	91	0.822	0.727	0.844	0.865	0.830	0.763	0.808
Fold-3	78	0.822	0.724	0.861	0.890	0.843	0.755	0.816
Fold-4	432	0.825	0.714	0.853	0.880	0.825	0.763	0.812
Fold-5	401	0.815	0.739	0.829	0.884	0.782	0.762	0.804
Fold-6	10	0.826	0.735	0.840	0.878	0.800	0.775	0.812
Fold-7	485	0.853	0.723	0.857	0.905	0.842	0.762	0.830
Mean Result	----	0.830	0.720	0.849	0.885	0.822	0.760	0.813

Table 4.5 Accuracy measures of information-gain method evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	144	0.851	0.754	0.840	0.877	0.836	0.782	0.825
Fold-2	126	0.853	0.779	0.846	0.895	0.840	0.783	0.833
Fold-3	143	0.862	0.766	0.839	0.898	0.825	0.782	0.834
Fold-4	179	0.851	0.764	0.850	0.891	0.846	0.781	0.832
Fold-5	321	0.862	0.773	0.846	0.895	0.838	0.789	0.836
Fold-6	442	0.861	0.771	0.850	0.900	0.837	0.787	0.837
Fold-7	169	0.848	0.762	0.853	0.888	0.838	0.788	0.831
Fold-8	529	0.870	0.775	0.828	0.888	0.810	0.798	0.834
Fold-9	149	0.862	0.776	0.840	0.889	0.829	0.793	0.833
Fold-10	284	0.856	0.751	0.858	0.898	0.845	0.782	0.835
Mean Result	----	0.858	0.767	0.845	0.892	0.835	0.786	0.833

Table 4.6 Accuracy measures of information-gain method evaluated by 7-fold cross validation experiment on EVAset

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	83	0.835	0.674	0.860	0.888	0.835	0.735	0.804
Fold-2	91	0.823	0.726	0.844	0.865	0.829	0.763	0.808
Fold-3	77	0.822	0.726	0.862	0.890	0.846	0.756	0.817
Fold-4	536	0.827	0.711	0.852	0.880	0.823	0.763	0.812
Fold-5	150	0.809	0.751	0.832	0.883	0.785	0.765	0.806
Fold-6	14	0.830	0.734	0.844	0.882	0.801	0.777	0.815
Fold-7	508	0.854	0.721	0.857	0.904	0.845	0.761	0.830
Mean Result	----	0.830	0.720	0.850	0.886	0.823	0.761	0.814

Table 4.7 Accuracy measures of gain ratio method evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	174	0.853	0.756	0.841	0.877	0.839	0.784	0.826
Fold-2	134	0.853	0.780	0.846	0.895	0.840	0.784	0.834
Fold-3	146	0.863	0.766	0.840	0.897	0.827	0.783	0.835
Fold-4	153	0.851	0.761	0.850	0.891	0.845	0.779	0.831
Fold-5	269	0.860	0.773	0.746	0.895	0.837	0.788	0.835
Fold-6	444	0.861	0.770	0.849	0.899	0.837	0.786	0.836
Fold-7	133	0.846	0.759	0.854	0.887	0.840	0.786	0.830
Fold-8	523	0.870	0.775	0.828	0.887	0.810	0.798	0.834
Fold-9	100	0.860	0.774	0.841	0.889	0.830	0.791	0.832
Fold-10	242	0.854	0.750	0.857	0.897	0.844	0.781	0.833
Mean Result	----	0.857	0.766	0.845	0.892	0.835	0.786	0.833

Table 4.8 Accuracy measures of gain ratio method evaluated by 7-fold cross validation experiment on EVAset

	Dimension	Recall ‘L’	Recall ‘H’	Recall ‘E’	Precision ‘L’	Precision ‘H’	Precision ‘E’	Accuracy
Fold-1	240	0.836	0.673	0.858	0.890	0.833	0.733	0.804
Fold-2	237	0.817	0.716	0.848	0.868	0.833	0.756	0.805
Fold-3	229	0.818	0.710	0.868	0.892	0.851	0.748	0.814
Fold-4	245	0.828	0.710	0.852	0.877	0.825	0.764	0.812
Fold-5	226	0.809	0.745	0.829	0.884	0.779	0.762	0.804
Fold-6	228	0.823	0.721	0.852	0.889	0.807	0.769	0.814
Fold-7	257	0.851	0.717	0.855	0.906	0.833	0.759	0.827
Mean Result	----	0.827	0.713	0.852	0.888	0.822	0.757	0.812

Table 4.9 Accuracy measures of genetic algorithm evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall ‘L’	Recall ‘H’	Recall ‘E’	Precision ‘L’	Precision ‘H’	Precision ‘E’	Accuracy
Fold-1	450	0.859	0.769	0.840	0.879	0.837	0.791	0.830
Fold-2	445	0.857	0.790	0.849	0.899	0.844	0.790	0.839
Fold-3	433	0.871	0.780	0.840	0.900	0.829	0.792	0.841
Fold-4	449	0.854	0.773	0.854	0.894	0.851	0.786	0.837
Fold-5	408	0.859	0.777	0.843	0.895	0.833	0.789	0.835
Fold-6	414	0.856	0.771	0.848	0.899	0.834	0.783	0.834
Fold-7	439	0.851	0.772	0.856	0.890	0.841	0.794	0.835
Fold-8	439	0.865	0.769	0.826	0.886	0.804	0.792	0.830
Fold-9	451	0.866	0.785	0.841	0.892	0.832	0.798	0.837
Fold-10	420	0.853	0.753	0.856	0.898	0.839	0.781	0.833
Mean Result	----	0.859	0.774	0.845	0.894	0.835	0.790	0.835

Table 4.10 Accuracy measures of genetic algorithm evaluated by 7-fold cross validation experiment on EVAset

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	16	0.836	0.669	0.852	0.879	0.833	0.732	0.800
Fold-2	16	0.822	0.724	0.837	0.855	0.827	0.762	0.804
Fold-3	15	0.826	0.708	0.856	0.879	0.844	0.752	0.811
Fold-4	16	0.825	0.718	0.838	0.860	0.816	0.766	0.806
Fold-5	14	0.806	0.748	0.825	0.877	0.780	0.761	0.801
Fold-6	16	0.826	0.728	0.840	0.879	0.794	0.774	0.811
Fold-7	15	0.849	0.717	0.839	0.894	0.819	0.756	0.820
Mean Result	----	0.828	0.716	0.841	0.876	0.816	0.758	0.808

Table 4.11 Accuracy measures of greedy algorithm evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	29	0.837	0.748	0.845	0.884	0.831	0.772	0.820
Fold-2	30	0.838	0.775	0.852	0.900	0.839	0.773	0.829
Fold-3	28	0.849	0.765	0.844	0.903	0.821	0.774	0.831
Fold-4	28	0.836	0.759	0.855	0.896	0.842	0.770	0.827
Fold-5	30	0.837	0.759	0.848	0.898	0.826	0.771	0.825
Fold-6	27	0.832	0.752	0.851	0.899	0.829	0.764	0.822
Fold-7	33	0.832	0.757	0.858	0.892	0.838	0.777	0.826
Fold-8	29	0.842	0.756	0.832	0.890	0.800	0.774	0.821
Fold-9	30	0.848	0.775	0.846	0.895	0.828	0.785	0.830
Fold-10	34	0.830	0.743	0.856	0.900	0.831	0.768	0.825
Mean Result	----	0.839	0.759	0.849	0.896	0.829	0.773	0.826

Table 4.12 Accuracy measures of greedy algorithm evaluated by 7-fold cross validation experiment on EVAset

	Dimension	Recall ‘L’	Recall ‘H’	Recall ‘E’	Precision ‘L’	Precision ‘H’	Precision ‘E’	Accuracy
Fold-1	16	0.836	0.669	0.852	0.879	0.833	0.732	0.800
Fold-2	16	0.822	0.724	0.837	0.855	0.827	0.762	0.804
Fold-3	15	0.826	0.708	0.856	0.879	0.844	0.752	0.811
Fold-4	16	0.825	0.718	0.838	0.860	0.816	0.766	0.806
Fold-5	16	0.811	0.749	0.826	0.879	0.778	0.765	0.804
Fold-6	16	0.826	0.728	0.840	0.879	0.794	0.774	0.811
Fold-7	15	0.849	0.717	0.839	0.894	0.819	0.756	0.820
Mean Result	----	0.839	0.716	0.841	0.876	0.815	0.759	0.808

Table 4.13 Accuracy measures of CFS and BestFirst search strategy evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall ‘L’	Recall ‘H’	Recall ‘E’	Precision ‘L’	Precision ‘H’	Precision ‘E’	Accuracy
Fold-1	34	0.838	0.749	0.844	0.882	0.833	0.773	0.821
Fold-2	30	0.838	0.775	0.852	0.900	0.839	0.773	0.829
Fold-3	33	0.851	0.764	0.844	0.902	0.822	0.774	0.831
Fold-4	31	0.836	0.758	0.855	0.896	0.842	0.770	0.827
Fold-5	30	0.837	0.759	0.848	0.898	0.826	0.771	0.825
Fold-6	34	0.835	0.753	0.850	0.897	0.831	0.765	0.823
Fold-7	32	0.832	0.756	0.858	0.892	0.838	0.777	0.826
Fold-8	29	0.842	0.756	0.832	0.890	0.800	0.774	0.821
Fold-9	33	0.849	0.775	0.846	0.895	0.828	0.786	0.831
Fold-10	34	0.837	0.743	0.856	0.900	0.831	0.768	0.825
Mean Result	----	0.840	0.759	0.849	0.896	0.829	0.773	0.826

Table 4.14 Accuracy measures of CFS and BestFirst search strategy evaluated by 7-fold cross validation experiment on EVAset

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	16	0.836	0.669	0.852	0.879	0.833	0.732	0.800
Fold-2	16	0.822	0.724	0.837	0.855	0.827	0.762	0.804
Fold-3	15	0.826	0.708	0.856	0.879	0.844	0.752	0.811
Fold-4	16	0.825	0.718	0.838	0.860	0.816	0.766	0.806
Fold-5	14	0.806	0.748	0.825	0.877	0.780	0.761	0.801
Fold-6	16	0.826	0.728	0.840	0.879	0.794	0.774	0.811
Fold-7	15	0.849	0.717	0.839	0.894	0.819	0.756	0.820
Mean Result	----	0.828	0.716	0.841	0.876	0.816	0.758	0.808

Table 4.15 Accuracy measures of minimum redundancy maximum relevance evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	23	0.833	0.748	0.845	0.885	0.829	0.770	0.819
Fold-2	24	0.837	0.776	0.852	0.901	0.839	0.773	0.829
Fold-3	24	0.847	0.764	0.846	0.904	0.822	0.772	0.830
Fold-4	25	0.834	0.757	0.856	0.898	0.842	0.768	0.827
Fold-5	25	0.834	0.757	0.849	0.900	0.825	0.768	0.823
Fold-6	25	0.832	0.752	0.851	0.900	0.830	0.763	0.822
Fold-7	27	0.829	0.757	0.858	0.892	0.836	0.775	0.825
Fold-8	26	0.844	0.753	0.832	0.890	0.799	0.774	0.821
Fold-9	26	0.847	0.772	0.846	0.896	0.827	0.783	0.829
Fold-10	27	0.835	0.742	0.856	0.901	0.828	0.767	0.824
Mean Result	----	0.837	0.758	0.849	0.897	0.828	0.771	0.825

Table 4.16 Accuracy measures of minimum redundancy maximum relevance evaluated by 7-fold cross validation experiment on EVAset

In the principal component analysis the number of dimensions is increased from 5 to 535 with increments of 5, and a one-versus-one SVM model is trained and tested on each train set for optimization and validation set respectively. Then, principal component analysis is applied in python pca library on train and test datasets for the optimum number of dimensions [67] . Finally, for each fold a one-versus-one SVM model is trained and tested by using the reduced datasets. The results of the 7-fold cross-validation experiment on CB513 are shown in

Table 4.17 and 10-fold cross-validation experiment on EVAset is presented in Table 4.18.

	Dimension	**Recall 'L'**	**Recall 'H'**	**Recall 'E'**	**Precision 'L'**	**Precision 'H'**	**Precision 'E'**	**Accuracy**
Fold-1	290	0.839	0.677	0.859	0.889	0.839	0.736	0.806
Fold-2	105	0.826	0.717	0.847	0.863	0.833	0.763	0.808
Fold-3	85	0.834	0.713	0.864	0.892	0.850	0.756	0.818
Fold-4	105	0.840	0.717	0.849	0.874	0.826	0.774	0.817
Fold-5	95	0.822	0.742	0.829	0.879	0.784	0.769	0.807
Fold-6	75	0.834	0.719	0.848	0.887	0.802	0.774	0.815
Fold-7	90	0.857	0.728	0.858	0.905	0.840	0.769	0.833
Mean Result	----	0.837	0.716	0.850	0.885	0.824	0.764	0.815

Table 4.17 Accuracy measures of principal component analysis evaluated by 7-fold cross validation experiment on CB513

	Dimension	**Recall 'L'**	**Recall 'H'**	**Recall 'E'**	**Precision 'L'**	**Precision 'H'**	**Precision 'E'**	**Accuracy**
Fold-1	590	0.863	0.764	0.839	0.879	0.836	0.792	0.830
Fold-2	470	0.860	0.778	0.846	0.898	0.837	0.787	0.836
Fold-3	550	0.871	0.775	0.837	0.900	0.822	0.790	0.839
Fold-4	380	0.855	0.760	0.851	0.894	0.841	0.782	0.833
Fold-5	430	0.858	0.760	0.843	0.894	0.826	0.783	0.830
Fold-6	560	0.860	0.756	0.846	0.898	0.830	0.780	0.831
Fold-7	570	0.855	0.762	0.849	0.891	0.827	0.791	0.831
Fold-8	550	0.866	0.774	0.824	0.887	0.800	0.795	0.831
Fold-9	590	0.867	0.776	0.837	0.890	0.826	0.795	0.834
Fold-10	580	0.856	0.743	0.852	0.897	0.832	0.779	0.830
Mean Result	----	0.861	0.765	0.843	0.893	0.828	0.787	0.833

Table 4.18 Accuracy measures of principal component analysis evaluated by 7-fold cross validation experiment on EVAset

For the autoencoder the number of hidden neurons, which gives the dimension of the reduced dataset, is increased from 75 to 525 with increments of 25. Maximum epoch number is set to 1000, L2WeightRegularization parameter to 0.004, SparsityRegularization parameter to 4, SparsityProportion parameter to

0.15 and scaleData parameter to false. As in other methods, autoencoder is applied in matlab [68] on train and test sets after finding the optimum number of dimensions and one-versus-one SVM is trained and tested on the reduced datasets. The results of the 7-fold cross-validation experiment on CB513 are shown in Table 4.19 and 10-fold cross-validation experiment on EVAset is given in Table 4.20.

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	175	0.827	0.620	0.882	0.897	0.860	0.718	0.799
Fold-2	225	0.824	0.674	0.865	0.863	0.862	0.754	0.811
Fold-3	275	0.828	0.659	0.876	0.891	0.862	0.743	0.813
Fold-4	275	0.830	0.697	0.866	0.877	0.847	0.771	0.820
Fold-5	225	0.823	0.710	0.852	0.878	0.826	0.767	0.817
Fold-6	250	0.834	0.684	0.867	0.893	0.826	0.771	0.819
Fold-7	275	0.851	0.684	0.873	0.908	0.861	0.759	0.831
Mean Result	----	0.832	0.675	0.869	0.888	0.849	0.755	0.820

Table 4.19 Accuracy measures of autoencoder evaluated by 7-fold cross validation experiment on CB513

	Dimension	Recall 'L'	Recall 'H'	Recall 'E'	Precision 'L'	Precision 'H'	Precision 'E'	Accuracy
Fold-1	450	0.857	0.737	0.849	0.881	0.844	0.777	0.832
Fold-2	375	0.821	0.717	0.862	0.899	0.833	0.752	0.819
Fold-3	425	0.831	0.736	0.853	0.892	0.827	0.763	0.823
Fold-4	425	0.841	0.707	0.845	0.882	0.829	0.755	0.819
Fold-5	450	0.847	0.761	0.851	0.903	0.821	0.778	0.832
Fold-6	400	0.838	0.720	0.856	0.895	0.840	0.755	0.825
Fold-7	425	0.838	0.754	0.856	0.897	0.845	0.766	0.831
Fold-8	450	0.853	0.730	0.853	0.893	0.836	0.768	0.830
Fold-9	425	0.841	0.719	0.856	0.892	0.842	0.759	0.825
Fold-10	450	0.809	0.744	0.864	0.900	0.841	0.752	0.818
Mean Result	----	0.837	0.733	0.855	0.893	0.836	0.762	0.825

Table 4.20 Accuracy measures of autoencoder evaluated by 7-fold cross validation experiment on EVAset

Training time of support vector machines for each fold is almost 7 hours for CB513 and almost 6 days for Evaset when all the features are used. That means total training time is almost 2 days for CB513 and almost 60 days for Evaset (assuming that these jobs are executed serially). Ranking methods consist of three different phases: ranking, finding the optimum number of features, and classification. For CB513 the total ranking time in each method is almost 5 hours, the total optimization time is almost 2 days and the total classification time is 15 hours. For EVAset the total ranking time is almost 4 days, total optimizing time is almost 250 days and total classification time is almost 22 days. For the remaining feature selection algorithms there are two phases: selection and classification. Except for the Genetic algorithm their running times are similar to each other. For CB513 feature selection takes 2 days and classification takes 5 hours. For EVAset feature selection takes 11 days and classification takes 29 days. In the CFS algorithm, which uses genetic algorithm as a search technique these times are longer than other feature selection algorithms. For CB513 feature selection takes 140 hours and classification takes almost 2 days. For Evaset selection takes 30 days and classification takes 55 days. Projection algorithms also have two phases: optimization and classification. In principal component analysis, optimization takes 7 days and classification takes 6 hours for CB513and for EVAset optimization takes 35 days and classification takes 50 days. In the deep autoencoder, for CB513 optimizing takes 6 days and classification takes 7 hours. For the Evaset set optimizing takes 35 days and classification takes 41 days. To decrease these times we sent each fold in parallel to different cores.

As shown in experiment results, feature selection and dimension reduction algorithms can be used to reduce the number of dimensions considerably for protein secondary structure prediction. For the two datasets (Evaset and CB513) the proposed autoencoder models and other models received similar accuracy. For CB513 deep autoencoder obtained the best overall accuracy (Q3) value of 0.820 and greedy, CFS and best first search strategy and minimum redundancy maximum relevance algorithms reduced the dimension the most with the mean number of dimensions equal to 15. Table 4.21 summarizes the percentage of eliminated attributes for CB513 and table 4.22 contains the percentage of eliminated attributes for EVAset. The best overall accuracy (Q3) for EVAset is obtained by CFS-Genetic search algorithm and the highest reduction in number of dimensions is achieved by CFS-MRMR algorithm with the mean number of dimensions acress the 10 folds equal to 25. The fastest algorithms (including optimization, selection and classification) are minimum CFS-MRMR, CFS-Best First and CFS-Greedy, the slowest algorithms are ranker feature selection algorithms for both two datasets.

	X^2	**IG**	**GR**	**Greddy**	**Genetic**	**CFS-Bestfist**	**MRMR**	**PCA**	**Autoencoder**
Fold-1	84.9	84.9	84.6	97.0	55.4	97.0	97.0	46.1	67.5
Fold-2	83.1	83.1	83.1	97.0	56.0	97.0	97.0	80.5	58.2
Fold-3	85.5	85.5	85.7	97.2	57.5	97.2	97.2	84.2	48.9
Fold-4	1.48	19.8	0.55	97.0	54.5	97.0	97.0	80.5	48.9
Fold-5	25.0	25.6	72.1	97.4	58.0	97.0	97.4	82.3	58.2
Fold-6	97.5	98.1	97.4	97.0	57.6	97.0	97.0	86.0	53.6
Fold-7	10.0	10.0	5.75	97.2	52.3	97.2	97.2	83.3	48.9
Mean Result	55,5	58,2	61,4	97,1	55,9	97,0	97,1	77,4	54,9

Table 4.21 Percentage of eliminated attributes for CB513

	X^2	IG	GR	Greddy	Genetic	CFS-Bestfist	MRMR	PCA	Autoencoder
Fold-1	83.9	84.5	81.3	96.8	51.6	96.3	97.5	36.6	51.6
Fold-2	87.0	86.4	85.6	96.7	52.2	96.7	97.4	49.5	59.7
Fold-3	86.0	84.6	84.3	96.9	53.4	96.4	97.4	40.9	54.3
Fold-4	80.2	80.7	83.5	96.9	51.7	96.6	97.3	59.1	54.3
Fold-5	66.8	65.5	71.1	96.7	56.1	96.7	97.3	53.8	51.6
Fold-6	54.7	52.5	52.3	97.0	55.5	96.3	97.3	39.8	57.0
Fold-7	80.1	81.8	85.7	96.4	52.8	96.5	97.0	38.7	54.3
Fold-8	42.1	43.1	43.8	96.8	52.8	96.8	97.2	40.9	51.6
Fold-9	87.6	83.9	89.2	96.7	51.5	96.4	97.2	36.6	54.3
Fold-10	69.2	89.4	74.0	96.3	54.8	96.3	97.0	37.7	51.6
Mean Result	74,0	75,4	75,3	96,7	53,2	96,5	97,2	43,4	54,0

Table 4.22 Percentage of eliminated attributes for EVAset

Chapter 5

Conclusions

In this thesis we employ deep autoencoder for dimension reduction and compare it with the traditional feature selection and dimension reduction techniques in protein secondary structure prediction on two benchmark datasets. In addition we compare the accuracy obtained after dimension reduction with the accuracy from the original feature set. As the classification method we use support vector machine, which is the second classifier of a two-stage predictor. As a result, feature selection and dimension reduction techniques achieved similar success rates compared to the accuracy obtained with the original feature set. They can be useful for protein structure prediction because they decrease the number of dimensions considerably. For each feature selection and projection algorithms, the classification phase is significantly than classification in original dimension. In addition some feature selection algorithms achieve better accuracy than the models trained using the original feature set. The proposed autoencoder model has similar success rate to the other models and it can be more ameliorative than the other models because of its several parameters. Furthermore, it takes the best accuracy value on the CB513 dataset and eliminates more than half of the features in both datasets. We can conclude that in most of the cases the autoencoder has better accuracy than other feature selection algorithms and projection methods except for the CFS-Genetic method. The disadvantages of the genetic algorithm are such that it takes longer time and cannot reduce the dimension considerably. Because of these reasons deep autoencoder is more useful as a dimension reduction algorithm. As a future work, we will apply the same methods to dihedral angle and solvent accessibility prediction, and analyze the improvement in accuracy and running times of the classification method.

BIBLIOGRAPHY

[1] J. Han, J. Pei, and M. Kamber, Data Mining: Concepts and Techniques. Elsevier, 2011.

[2] S. Wold, K. Esbensen, and P. Geladi, ‘Principal component analysis’, Chemom. Intell. Lab. Syst., vol. 2, no. 1, pp. 37–52, Aug. 1987.

[3] G. E. Hinton, M. Revow, and P. Dayan, ‘Recognizing Handwritten Digits Using Mixtures of Linear Models’, in Proceedings of the 7th International Conference on Neural Information Processing Systems, Cambridge, MA, USA, 1994, pp. 1015–1022.

[4]T. Kavzoğlu, E. K. Şahin, and İ. Çölkesen, ‘Heyelan Duyarlılık Analizinde Ki-Kare Testine Dayalı Faktör Seçimi’, presented at the V. Uzaktan Algılama ve Coğrafi Bi lgi Sistemleri Sempozyumu (UZAL CBS 2014), 2014.

[5] P. Ozarkar and M. Patwardhan, ‘Efficient Spam Classification by Appropriate Feature Selection’, Glob. J. Comput. Sci. Technol. Softw. Data E Ngineering, vol. 1 3, no. 5, 2013.

[6] K. O. Jeppson, ‘Modeling the influence of the transistor gain ratio and the input-to-output coupling capacitance on the CMOS inverter delay’, IEEE J. Solid-State Circuits, vol. 29, no. 6, pp. 646–654, Jun. 1994.

[7] C. Ding and H. Peng, ‘Minimum redundancy feature selection from microarray gene expression data’, J. Bioinform. Comput. Biol., vol. 03, no. 02, pp. 185–205, Nisan 2005.

[8] W. Siedlecki and J. Sklansky, ‘A note on genetic algorithms for large-scale feature selection’, Pattern Recognit. Lett., vol. 10, no. 5, pp. 335–347, Nov. 1989.

[9] N. Kwak and C.-H. Choi, ‘Input feature selection for classification problems’, IEEE Trans. Neural Netw., vol. 13, no. 1, pp. 143–159, Jan. 2002.

[10] L. Xu, P. Yan, and T. Chang, ‘Best first strategy for feature selection’, in [1988 Proceedings] 9th International Conference on Pattern Recognition, 1988, pp. 706–708 vol.2.

[11] ‘Protein yapısı’, http://tr.wikipedia.org/wiki/Protein_yapısı. (Jun 2017)

[12] ‘Protein - Primary, Secondary, Tertiary and Quaternary structure’, http://biology4alevel.blogspot.com.tr/2014/08/12-protein-primary-secondary-tertiary.html. (Jun 2017)

[13] ‘Alfa sarmal’, http://tr.wikipedia.org/wiki/Alfa_sarmal. (Jun 2017)

[14] ‘Difference Between Alpha Helix and Beta Pleated Sheet’, http://pediaa.com/difference-between-alpha-helix-and-beta-pleated-sheet/. (Jun 2017)

[15] ‘Beta yaprak’, http://tr.wikipedia.org/wiki/Beta_yaprak. (Jun 2017)

[16] ‘Structure,Classification and Function Of Protein’, http://www.biologynoteshelp.com/primary-structuresecondary-structure-of-protein/. (Jun 2017)

[17] ‘Protein structure prediction’, http://en.wikipedia.org/wiki/Protein_structure_prediction. (Jun 2017)

[18] Z. Aydin, A. Singh, J. Bilmes, and W. S. Noble, ‘Learning sparse models for a dynamic Bayesian network classifier of protein secondary structure’, BMC Bioinformatics, vol. 12, p. 154, 2011.

[19] C. Mirabello and G. Pollastri, ‘Porter, PaleAle 4.0: high-accuracy prediction of protein secondary structure and relative solvent accessibility’, Bioinformatics, vol. 29, no. 16, pp. 2056–2058, Aug. 2013.

[20] G. Pollastri, A. J. Martin, C. Mooney, and A. Vullo, ‘Accurate prediction of protein secondary structure and solvent accessibility by consensus combiners of sequence and structure information’, BMC Bioinformatics, vol. 8, p. 201, 2007.

[21] D. Li, T. Li, P. Cong, W. Xiong, and J. Sun, ‘A novel structural position-specific scoring matrix for the prediction of protein secondary structures’, Bioinformatics, vol. 28, no. 1, pp. 32–39, Jan. 2012.

[22] A. A. Salamov and V. V. Solovyev, ‘Prediction of Protein Secondary Structure by Combining Nearest-neighbor Algorithms and Multiple Sequence Alignments’, J. Mol. Biol., vol. 247, no. 1, pp. 11–15, Mar. 1995.

[23] D. T. Jones, ‘Protein secondary structure prediction based on position-specific scoring matrices1’, J. Mol. Biol., vol. 292, no. 2, pp. 195–202, Eylül 1999.

[24] L. Jian-wei, C. Guang-hui, L. Hai-en, L. Yuan, and L. Xiong-lin, ‘Prediction of protein secondary structure using multilayer feed-forward neural networks’, in 2013 25th Chinese Control and Decision Conference (CCDC), 2013, pp. 1346–1351.

[25] A. Yaseen and Y. Li, ‘Context-Based Features Enhance Protein Secondary Structure Prediction Accuracy’, J. Chem. Inf. Model., vol. 54, no. 3, pp. 992–1002, Mar. 2014.

[26] X.-Q. Yao, H. Zhu, and Z.-S. She, ‘A dynamic Bayesian network approach to protein secondary structure prediction’, BMC Bioinformatics, vol. 9, p. 49, 2008.

[27] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi, ‘Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles’, Proteins Struct. Funct. Bioinforma., vol. 47, no. 2, pp. 228–235, May 2002.

[28] A. Ghosh and B. Parai, ‘Protein secondary structure prediction using distance based classifiers’, Int. J. Approx. Reason., vol. 47, no. 1, pp. 37–44, Ocak 2008.

[29] W. Yang, K. Wang, and W. Zuo, ‘A fast and efficient nearest neighbor method for protein secondary structure prediction’, in 2011 3rd International Conference on Advanced Computer Control, 2011, pp. 224–227.

[30] S. Hua and Z. Sun, ‘A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach1’, J. Mol. Biol., vol. 308, no. 2, pp. 397–407, Nisan 2001.

[31] Y. F. Huang and S. Y. Chen, ‘Protein secondary structure prediction based on physicochemical features and PSSM by SVM’, in 2013 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2013, pp. 9–15.

[32] Y. Wang, J. Cheng, Y. Liu, and Y. Chen, ‘Prediction of protein secondary structure using support vector machine with PSSM profiles’, in 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, 2016, pp. 502–505.

[33] J. Martin, J.-F. Gibrat, and F. Rodolphe, ‘Analysis of an optimal hidden Markov model for secondary structure prediction’, BMC Struct. Biol., vol. 6, p. 25, 2006.

[34] Z. Aydin, Y. Altunbasak, and M. Borodovsky, ‘Protein secondary structure prediction for a single-sequence using hidden semi-Markov models’, BMC Bioinformatics, vol. 7, p. 178, 2006.

[35] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, ‘Extreme learning machine: Theory and applications’, Neurocomputing, vol. 70, no. 1–3, pp. 489–501, Aralık 2006.

[36] G. Wang, Y. Zhao, and D. Wang, ‘A protein secondary structure prediction framework based on the Extreme Learning Machine’, Neurocomputing, vol. 72, no. 1–3, pp. 262–268, Aralık 2008.

[37] L. Lin, S. Yang, and R. Zuo, ‘Protein secondary structure prediction based on multi-SVM ensemble’, in 2010 International Conference on Intelligent Control and Information Processing, 2010, pp. 356–358.

[38] H. Bouziane, B. Messabih, and A. Chouarfia, ‘Effect of simple ensemble methods on protein secondary structure prediction’, Soft Comput., vol. 19, no. 6, pp. 1663–1678, Jun. 2015.

[39] M. Spencer, J. Eickholt, and J. Cheng, ‘A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction’, IEEE/ACM Trans. Comput. Biol. Bioinform., vol. 12, no. 1, pp. 103–112, Ocak 2015.

[40] Z. Li and Y. Yu, ‘Protein Secondary Structure Prediction Using Cascaded Convolutional and Recurrent Neural Networks’, ArXiv160407176 Cs Q-Bio, Apr. 2016.

[41] S. Wang, J. Peng, J. Ma, and J. Xu, ‘Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields’, Sci. Rep., vol. 6, Jan. 2016.

[42] Z. Li, J. Wang, S. Zhang, Q. Zhang, and W. Wu, ‘A new hybrid coding for protein secondary structure prediction based on primary structure similarity’, Gene, vol. 618, pp. 8–13, Haziran 2017.

[43] R. Adamczak, ‘Dimensionality reduction of PSSM matrix and its influence on secondary structure and relative solvent accessibility predictions’, presented at the World Academy of Science, Engineering and Technology 58, 2009.

[44] M. H. Zangooei and S. Jalili, ‘Protein secondary structure prediction using DWKF based on SVR-NSGAII’, Neurocomputing, vol. 94, pp. 87–101, Ekim 2012.

[45] S. Fayech, N. Essoussi, and M. Limam, ‘Data mining techniques to predict protein secondary structures’, in 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), 2013, pp. 1–5.

[46] S. Altschul, ‘Gapped BLAST and PSI-BLAST: a new generation of protein database search programs’, Nucleic Acids Res., vol. 25, no. 17, pp. 3389–3402, Sep. 1997.

[47]‘BLAST+ executables’, http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download. (Jun 2017)

[48] M. Remmert, A. Biegert, A. Hauser, and J. Söding, ‘HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment’, Nat. Methods, vol. 9, no. 2, pp. 173–175, ubat 2012.

[49] ‘HH-suite download’, ftp://toolkit.genzentrum.lmu.de/pub/HH-suite/. (Jun 2017)

[50] T. Zhou, N. Shu, and S. Hovmöller, ‘A novel method for accurate one-dimensional protein structure prediction based on fragment matching’, Bioinformatics, vol. 26, no. 4, pp. 470–477, Feb. 2010.

[51] Z. Aydin, D. Baker, and W. S. Noble, ‘Constructing Structural Profiles for Protein Torsion Angle Prediction’:, 2015, pp. 26–35.

[52] ‘RCSB Protein Data Bank - RCSB PDB’, https://www.rcsb.org/pdb/home/home.do. (Jun 2017)

[53] V. Vapnik, The Nature of Statistical Learning Theory. Springer Science & Business Media, 2013.

[54]‘BestFirst’, http://weka.sourceforge.net/doc.dev/weka/attributeSelection/BestFirst.html. (Jun 2017)

[55] V. Dang and W. B. Croft, ‘Feature Selection for Document Ranking Using Best First Search and Coordinate Ascent’, presented at the Sigir workshop on feature generation and selection for information retrieval, 2010.

[56] P. Baldi, ‘Autoencoders, Unsupervised Learning, and Deep Architectures’, presented at the Workshop on Unsupervised and Transfer Learning, 2012.

[57] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, ‘Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1’, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362.

[58]'Autoencoders, Tied Weights and Dropout', http://image.diku.dk/shark/sphinx_pages/build/html/rest_sources/tutorials/algorithms/autoencoders.html. (Jun 2017)

[59] 'Denoising Autoencoders' http://deeplearning.net/tutorial/dA.html. (Jun 2017)

[60] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, 'Extracting and Composing Robust Features with Denoising Autoencoders', in Proceedings of the 25th International Conference on Machine Learning, New York, NY, USA, 2008, pp. 1096–1103.

[61] J. A. Cuff and G. J. Barton, 'Evaluation and improvement of multiple sequence methods for protein secondary structure prediction', Proteins Struct. Funct. Bioinforma., vol. 34, no. 4, pp. 508–519, Mar. 1999.

[62] I. Y. Y. Koh, 'EVA: evaluation of protein structure prediction servers', Nucleic Acids Res., vol. 31, no. 13, pp. 3311–3315, Jul. 2003.

[63] 'Precision and recall', https://en.wikipedia.org/wiki/Precision_and_recall. (Jun 2017)

[64] 'Cross-validation (statistics)', https://en.wikipedia.org/wiki/Cross-validation_(statistics). (Jun 2017)

[65] 'DSSP', http://swift.cmbi.ru.nl/gv/dssp/ (Jun 2017)

[66]'Performing attribute selection', http://weka.wikispaces.com/Performing%20attribute%20selection (Jun 2017)

[67]'Principal components analysis (PCA)', http://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_3d.html (Jun 2017)

[68]'trainAutoencoder', https://www.mathworks.com/help/nnet/ref/trainautoencoder.html (Jun 2017)