Ömmu Gülsüm UZUT — OPTIMIZING CLASSIFIERS FOR PROTEIN SECONDARY STRUCTURE PREDICTION — AGU 2017

OPTIMIZING CLASSIFIERS 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'S

By
Ömmu Gülsüm UZUT
July 2017

OPTIMIZING CLASSIFIERS FOR PROTEIN SECONDARY STRUCTURE PREDICTION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF NATURAL SCIENCES OF
ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER'S

By
Ömmu Gülsüm UZUT
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: Ömmu Gülsüm UZUT

Signature :

REGULATORY COMPLIANCE

M. Sc. thesis titled Optimizing Classifiers 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

Ömmu Gülsüm UZUT

Advisor

Assist. Prof. Zafer AYDIN

Head of the Electrical and Computer Engineering Program

Assoc. Prof. Vehbi Çağrı GÜNGÖR

ACCEPTANCE AND APPROVAL

M. Sc. thesis titled Optimizing Classifiers for Protein Secondary Structure Prediction and prepared by Ömmu Gülsüm UZUT 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

Member : Prof. Bülent YILMAZ

Member : Assist. Prof. Ufuk NALBANTOĞLU

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

OPTIMIZING CLASSIFIERS FOR PROTEIN SECONDARY STRUCTURE PREDICTION

Ömmu Gülsüm UZUT
Master's program in Electrical and Computer Engineering Department
Supervisor: Assist. Prof. Zafer AYDIN

July 2017

Protein secondary structure prediction (PSSP) is important for understanding protein structure and function. It can be seen as a bridge between amino acid sequence and three-dimensional (3-D) structure of a protein. To date, many methods have been proposed to improve prediction accuracy. There are multiple conditions that will affect the performance of a method. One of these is the selection of correct hyper parameters, which may not be learned directly from the regular training process. Optimizing these hyper-parameters enable us to fine-tune the model complexity preventing over-fitting and under-fitting.

In this thesis, we optimized a support vector machine, a deep convolutional neural field and a random forest for the second stage of a hybrid classifier for protein secondary structure prediction. In addition, we built an ensemble classifier that combines the predictions from the individual methods in various combinations. We demonstrate that the overall accuracy of the ensemble is comparable to the success rates of the state-of-the-art methods in the most difficult prediction setting and combining the selected models have the potential to further improve the accuracy of the base learners.

Keywords: Bioinformatics, Protein Secondary Structure Prediction, Ensemble Methods, Hybrid Classifiers, Deep Learning.

ÖZET

PROTEİN İKİNCİL YAPISININ TAHMİNİ İÇİN SINIFLANDIRMA YÖNTEMLERİNİN OPTİMİZASYONU

Ömmu Gülsüm UZUT
Elektrik ve Bilgisayar Mühendisliği Ana Bilim Dalı, Yüksek Lisans Programı
Tez Yöneticisi: Yrd. Doç. Dr. Zafer AYDIN

Temmuz 2017

Protein ikincil yapı tahmini, proteinin yapısını ve fonksiyonunu anlamak için önemli ve yaygın olarak kullanılan bir aşamadır. İkincil yapı tahmin bilgisi üç boyutlu yapı tahmini için de kullanıldığından protein dizisiyle üç boyutlu yapısı arasında bir köprü olarak görülebilir. Şimdiye kadar, tahmin doğruluk oranını artırmak için birçok yöntem geliştirilmiştir. Yöntemlerin performansını etkileyecek birden fazla durum vardır. Bunlar arasında model hiper-parametrelerinin doğru seçilmesi önem taşımaktadır. Model eğitme sürecinde direkt olarak öğrenilemeyen bu parametrelerin optimize edilmesiyle modellerin hassas olarak ayarlanması mümkündür. Bu sayede aşırı uyum ve eksik uyum gibi davranışlardan kaçınılması amaçlanır.

Bu tezde, destek vektör makinesi, derin katlamalı yapay sinir alanları ve rastgele orman yöntemleri bir hibrit sınıflandırıcının ikinci aşamasında kullanılmak üzere optimize edilmiş ve ikincil yapı tahmini problemine uygulanmıştır. Buna ek olarak eğitilen sınıflandırıcılardan elde edilen tahminler bir topluluk yöntemi ile farklı kombinasyonlarda birleştirilmiş ve başarı oranları en zor tahmin koşulu için incelenmiştir. Geliştirilen yöntemlerin doğruluk oranları literatürdeki en iyi yöntemler ile aynı seviyededir ve farklı modellerin birleştirilmesinin tahmin başarısını iyileştirme potansiyeli bulunduğu gösterilmiştir.

Anahtar Kelimeler: Biyoenformatik, Protein İkincil Yapı Tahmini, Hibrit Sınıflandırıcılar, Topluluk Yöntemleri, Derin Öğrenme

Acknowledgements

I would like to express my deepest regards to my advisor Assistant Professor Zafer AYDIN, for his interest and invaluable helpfulness. I am so grateful to study with him.

I would like to thank my family especially my father Zülküf UZUT and my mother Nisbet UZUT for being in favor of me in any case and believing in me.

All computations were performed on TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA Resources). This work is supported by grant 113E550 from 3501 TUBITAK National Young Researchers Career Award.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This thesis is dedicated to my family

Chapter 1

1.Introduction

Using computational techniques has gained widespread attention in bioinformatics because of the exponential growth, complexity and accessibility of biological data. On the path toward discovering the knowledge hidden in this large amounts of data, machine learning approaches has played an important role. It has been successfully applied to several problems including protein structure prediction [1], protein sequence analysis [2], protein fold recognition [3, 4, 5], protein function prediction [6], gene network inference [7], metabolic pathway analysis [8].

Prediction of three dimensional structure of a protein from its amino acid sequence is known as tertiary structure prediction [9], which has been one of the most challenging problems in bioinformatics. Three dimensional (3D) structure of a protein gives crucial information about its function. Despite being accurate, solving the structure through experiments is expensive and time consuming. In this respect, protein tertiary structure provides an alternative to experimental techniques. Furthermore the 3D structure information can also be used for designing new drugs. Due to the challenging nature of the problem instead of solving it directly first various structural properties of proteins are estimated such as sequence profiles, solvent accessibility, dihedral angles, and contact maps.

Protein secondary structure is formed by regular hydrogen bonding patterns that stabilize the protein structure [10]. Protein secondary structure prediction (PSSP) aims to assign a structural state from a three letter alphabet, which includes H for helix, E for strand and L for loop. To predict secondary structure, typically supervised learning is used, in which a model is trained using proteins with known secondary structure labels derived from 3D structure information available in Protein Data Bank (PDB) [11]. Various methods have been proposed for secondary structure prediction. Among those

support vector machines (SVM) and neural networks stand out with promising accuracy values as compared to other methods. Ward et al. applied support vector machines to 3-state protein secondary structure prediction on a set of 1460 proteins and obtained a 77.07 % accuracy by cross-validation [12]. Hua and Sun showed that it is feasible to get improvement by adding PSI-BLAST generated profiles as input features to SVMs [13]. Kim and Park developed a new method, SVMpsi, to improve the prediction rate using PSI-BLAST PSSM profiles. The method obtained 4.9% and 3.1% improvements on RS126 [14] and CB513 [15] datasets, respectively [16]. Gubbi and Lai applied an SVM to protein secondary structure prediction and obtained a 77.9% accuracy by seven fold cross-validation on CB513 [17]. Also there are additional studies, in which SVM outperforms as compared to other machine learning classifiers [18].

Ensemble learning is an important technique in pattern recognition, machine learning and data mining. The main idea behind ensemble learning is to combine multiple classifiers for improving the accuracy rate [19].

Recently, many studies have been performed to improve the accuracy through combining different methods. King used ensemble methods that combine several machine learning approaches by voting and obtained a better accuracy than individual classifiers on CASP dataset [20]. Kountouris also combined machine learning techniques for secondary structure prediction and showed that the resulting method can improve the quality of the predictions, especially the SOV score [21]. Alirezaa also used a machine learning approach which included an ensemble of neural networks with different voting combination methods for class imbalance problem of protein secondary structure prediction and showed their ensemble system has better performance when compared with the individual classifier they employed [22]. Pollastri and Baldi have studied on ensembles of bidirectional recurrent neural network to improve contact and accessibility prediction [23]. There are additional studies that use ensemble methods and show improvement over the performance of individual classifiers [24-29]. Besides ensembles, there are also hybrid methods that combine the strengths of various classifiers. Yao et al. introduced a two-stage classifier that employs Dynamic Bayesian Networks (DBNs) and neural networks for protein secondary structure prediction [30]. Aydin et al. improved this model by incorporating features derived from HMM-profiles, sparsifying DBN models and employing an SVM classifier instead of neural networks, which provided state-of-the-art performance [31]. Recently Peng et al. proposed a deep conditional neural fields model that combines a deep convolutional network with a

conditional random field, which are trained jointly [32,33]. Based on these advancements, it is of interest to analyze how well the SVM and deepCNF models complement each other and whether combining predictions obtained from these methods improves the accuracy.

In this thesis, we optimized the hyper-parameters of three classification methods: a support vector machine, a deepCNF and a random forest, which are employed at the second stage of the hybrid classifier introduced in Aydin et al. [31]. We then implemented an ensemble that combines the predictions of these models in various combinations by averaging the probability distributions of class labels.

This thesis is organized as follows. Chapter 2 gives information about proteins and their structures. Pre-processing methods about preparing feature vectors are elucidated in chapter 3, which also includes the methods used for parameter optimization and prediction. Chapter 4 includes our results and conclusions of the thesis are provided in Chapter 5.

Chapter 2

2. Structure of a Protein

Proteins are polymers of amino acids consisting of one or more polypeptide chains. “Proteins perform a vast array of functions within organisms, including catalysing metabolic reactions, DNA replication, responding to stimuli, and transporting molecules from one location to another. Proteins differ from one another primarily in their sequence of amino acids, which is dictated by the nucleotide sequence of their genes, and which usually results in protein folding into a specific three-dimensional structure that determines its activity.”[34]

Despite having the same general structure, the side chain (R group) of each amino acid is different. There are twenty types of amino acids commonly found in nature, which have different physical and chemical characteristics such as electrostatic charge, acid separation coefficient, hydrophobicity, size and functional group. These properties play an important role in determining the structure of a protein [35].

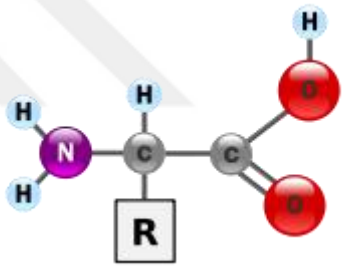


Figure 2.1 The structure of an amino acid [35].

2.1 Protein Structure Levels

There are four basic levels of protein structure: primary, secondary, tertiary and quaternary. Primary structure is the amino acid sequence of a protein. Secondary structure is formed through regular hydrogen bonds. Tertiary structure is the three dimensional structure of an amino acid chain. Quaternary structure is the three dimensional structure of multiple amino acid chains that form a protein.

2.1.1 Primary Structure

The primary structure of a protein consists of all the necessary information for determining the 3D structure. The alteration of one amino acid in the sequence can change the entire protein.

Primary structure is important because

- many genetic diseases occur due to abnormal amino acid sequences
- it serves as a starting point for predicting secondary and tertiary structure.
- it conveys information about the molecular system of proteins

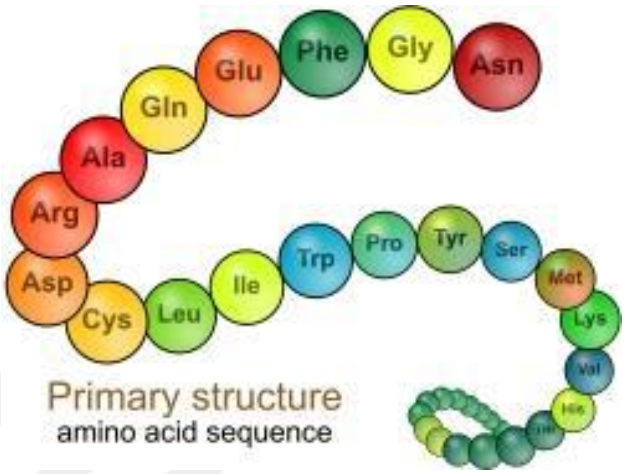


Figure 2.1.1.1 Primary structure of an amino acid sequence [35].

2.1.2 Secondary Structure

Secondary structure of a protein is the local structural conformation that is formed by regular hydrogen bonding patterns that stabilize the coiling and folding of polypeptide chains. There are two regular and major structural elements of secondary structure. These are α helix and β strand.

Coiling occurs, forming an **α helix**, due to the formation of repetitive hydrogen bonds between the nitrogen of one amino acid and the oxygen of another located in neighboring part of the polypeptide chain. The amino acids in α helix are usually arranged in a right-handed helical structure and each helix contains from 5 to 40 amino acids.

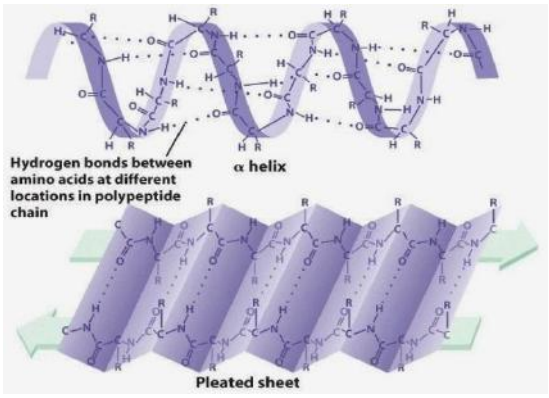


Figure 2.1.2.1 Secondary structure of an amino acid sequence [31]

β pleated sheets are formed due to hydrogen bonding between different amino acid segments (peptides) arranged side by side.

There are two types of pleated sheets: parallel and antiparallel. If two

peptides run in the same direction, it is called parallel β pleated sheets. For antiparallel β pleated sheets, the peptides must run in opposite directions.

Loops are not regular structures in proteins unlike alpha helices and beta sheets. They are typically formed in between the helix and beta sheets and usually found at the surface of the protein. Loops contain turns, random coils and bends details of which are explained in [10].

2.1.3 Tertiary Structure

Tertiary structure is the three dimensional coordinates of the atoms in an amino acid chain. Every protein has a unique three dimensional structure and the function of the protein depends closely on its structure. Understanding the function and structure opens the doors for diagnosing diseases, designing drugs and investigating new treatment models.

The 3D structure is typically determined by X-ray crystallography and NMR (Nuclear Magnetic Resonance) spectroscopy. The atomic coordinates of the solved structures are collected in a database known as the Protein Data Bank (PDB) [11]. Despite providing accurate information about the structure of proteins, these methods are expensive and time consuming. Therefore, computational methods are applied as an alternative solution technique.

2.1.4 Quaternary Structure

Quaternary structure is the combination of two or more amino acid chains. It is stabilized by non-covalent and disulfide bonds which also stabilize tertiary structures.

2.2 Protein Structure Prediction

Protein structure prediction is an important problem in bioinformatics due to relation between the function of a protein and its three dimensional structure. Although a lot of research has been performed on protein structure prediction, it has not been solved completely.

2.2.1 Secondary Structure Prediction

Secondary structure prediction aims to assign a secondary structure class label (H: helix, E: strands, L: loop) to each amino acid of a protein (Figure 2.2.1.1).

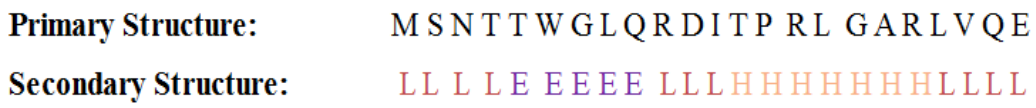


Figure 2.2.1.1 Three state secondary structure prediction. The first line represents the amino acid sequence, the second line shows the secondary structure labels (H: coil, E: beta strand L: loop).

For secondary structure prediction, generally supervised learning approaches are used. In supervised learning, a model is trained from the database with known secondary structure labels to make prediction for proteins with unknown structure.

2.2.2 Secondary Structure Types

2.2.2.1 8-state representation

DSSP (dictionary of secondary structure prediction) is a program that defines secondary structure of a protein starting from the 3D coordinate information. DSSP uses an eight state representation denoted by single letter codes [10]. There are also other programs that extract secondary structure labels starting from 3D coordinates of the atoms such as DEFINE [37] and STRIDE [38].

H	α - helix
G	3_{10} – helix
I	π - helix (extremely rare)
E	β - strand
B	β - bridge
T	β- turn
S	Bend
L	the rest

Table 2.2.2.1.1 8 class representation of protein secondary structure.

Although the original definition of secondary structure contains eight states prediction methods are generally trained and evaluated with three states due to due to scarcity of data in certain classes and similarity between classes that belong to the same structural group. There are different conventions to map eight state representation to three states. In this thesis we used the following mapping: H, G, I → H; E, B → E; S, T, ' ' → L, in which H refers to helix, E refers to strand and L refers to loop. This is the most widely used and the most difficult mapping. Other transformations include

- H, G → H; E, B → E; I, S, T, ' ' → L
- H, G → H; E → E; B, I, S, T, ' ' → L
- H → H; E → E; G, B, I, S, T, ' ' → L

2.3 Measures for Prediction Accuracy

The overall accuracy denoted by Q_3 is the most popular measure in the literature when evaluating performance of secondary structure prediction methods. It is defined as the percentage of the amino acids that are correctly predicted to be one of the three states (H, E, L).

$$Q_3 = \frac{100x(TP_H + TP_E + TP_L)}{N} \quad \textbf{(2.3.1)}$$

where TP_H is the number of true positives for helix, TP_E is the number of true positives number for strand, TP_L is the number of true positives number for loop and N is the total number of amino acids. Similar to equation 2.3.1 we can define the accuracy of each class type as follows

$$Q_{CL} = \frac{100xTP_{CL}}{N_{CL}} \quad \textbf{(2.3.2)}$$

where Q_{CL} is the percentage of correctly predicted amino acids that belong to class CL with $CL \in \{H, E, L\}$, TP_{CL} is number of true positives for CL and N_{CL} is the number of amino acids in state CL.

In addition to the above there are also other measures used for evaluation such as the segment overlap (SOV) that is used for testing the average overlap between the observed and the predicted segments rather than individual residues [39] and Matthew's Correlation Coefficient (MCC) [40].

Chapter 3

3. Methods

Analyzing large quantities of data manually is not feasible. At this stage, machine learning methods can be used to discover and learn patterns in data and make predictions for new data.

In this thesis, we used the DSPRED method introduced in Aydin et al. [31], which is a hybrid classifier that combines dynamic Bayesian networks and a support vector machine (SVM) for predicting the secondary structure of proteins. We considered replacing the SVM with deep convolutional neural field [33] and random forest. Additionally we analyzed the effect of combining predictions from the three classifiers in an ensemble framework.

3.1 Dataset

We used the CB513 benchmark [15] dataset that contains 513 proteins and 84,119 amino acid residues. It was obtained by combining 396 sequences in CB396 benchmark and the 117 sequences in RS126 [14] after removing the duplicates. CB513 is a popular benchmark used in protein secondary structure prediction.

3.2 Feature Extraction

In this thesis, we employ two types of input features: position specific scoring matrices (PSSM) derived from sequence alignments and structural profile matrices. The PSSMs are computed by PSI-BLAST [41] and HHBlits [42] programs and are named as PSI-BLAST PSSM and HHMAKE PSSM, which are used as input features for the DBN models as well as the classifiers in the second stage of the hybrid method. The

structural profile matrices are computed after the second stage of the HHBlits program using the structure labels of the PDB proteins aligned to the target.

3.2.1 PSSM

A position specific scoring matrix (PSSM) contains scores summarizing the statistical characteristics of proteins in a family, which are assumed to have similar structure and function. Each column represents an amino acid position and contains the likelihood of observing the twenty amino acids at that position among proteins that belong to the same family. A PSSM can be constructed by aligning a query protein against a database of proteins, jointly aligning the hits that score above threshold using multiple alignment, and computing a weighted frequency of occurrence for the twenty amino acids at each position of the alignment. Following this procedure a different PSSM can be obtained for each query protein due to slight differences in the alignments between proteins in the same family. Therefore, a PSSM can be used as input features for structure prediction methods because it serves as a signature for a query protein summarizing the statistics of sequence-based similarities against the proteins in the same family.

3.2.2 PSI-BLAST PSSM

PSI-BLAST is an iterative method that searches the database of sequences to find proteins that are distantly related to the query. In the first iteration, it performs a regular BLAST search and aligns the query to database proteins by pairwise alignment. Then it performs a multiple alignment and builds a profile matrix (i.e. PSSM) using proteins with scores above threshold. In the second and subsequent iterations, the profile matrix is aligned to the proteins in the database to discover more distant proteins (i.e. those with little sequence similarity but high structural and functional similarity). After this alignment proteins that score above threshold are multiply aligned and the profile matrix is updated. The procedure stops after reaching a certain number of iterations, which is a parameter set by the user. At the end a profile matrix of size $20 * N$ is generated, where N is the number of amino acids in the target protein. Because of being fast and simple to implement, possibility to search PSSMs on large database, providing efficiency and sensitivity, PSI-BLAST is one of the most commonly used profile matrix derivation method for structure prediction. However, PSI-BLAST can also perform

mismatches (i.e. false positives). Therefore the profile matrices derived from this method contain a certain noise. Nonetheless, the first profile matrix used in this thesis is obtained by the PSI-BLAST method due to its popularity and having a certain level of accuracy. PSI-BLAST PSSMs are computed by aligning the sequences in CB513 dataset against the NR database. This is followed by scaling, in which the scores are normalized by a sigmoidal transformation.

3.2.3 HHMAKE PSSM

Hidden Markov models (HMMs) are mainly used as a classifier in handwriting recognition, speech recognition [43], and in bioinformatics problems such as protein secondary structure prediction [30] and protein torsion angle prediction [44]. In bioinformatics HMMs can also be used to compute profiles to represent proteins in the same family. Profiles based on hidden Markov models (HMM-profiles) can be obtained after computing multiple alignments with the proteins found by sequence alignment algorithms. Similar to PSI-BLAST these HMM-profiles can be used iteratively for profile sequence alignment or profile-profile alignment [42] to discover distantly related proteins. HMM profiles are more sensitive than standard profiles and explore more distant protein similarities. One of the best software for computing HMM-profiles is HHBlits [42], which is faster than PSI-BLAST due to recent improvements on the speed of the original algorithm.

In this thesis, HHMAKE PSSM features are derived from HMM-profile models computed by the first step of the HHBlits method. For this task, each target in CB513 is aligned against the NR20 database and hit proteins that score above the threshold are jointly aligned using a multiple alignment algorithm. Then an HMM-profile model is constructed starting from the multiple alignment. Finally the weighted frequencies in each match state of the HMM–profile are normalized by min-max scaling (i.e. linear transformation) into the interval [0,1] to obtain the HHMAKE PSSM.

3.2.4 Structural Profiles

In addition to profile matrices based on multiple alignments of amino acid sequences for structural prediction, structural profile matrices have also been used as attributes. They are constructed using the structural sequences of the proteins found by sequence alignment methods. The dimension of a structural profile matrix formed for the secondary structure estimation is $3 * N$, where N is the number of amino acids in the

target protein and each column has the weighted frequency scores of the three secondary structure states for an amino acid. Fig. 3.2.4.1 shows an example structural profile matrix.

	1	2	…	N
H	0.4	0.5	…	0.2
E	0.3	0.5	…	0.3
L	0.3	0	…	0.5

Figure 3.2.4.1 A structural profile for 3 state secondary structure prediction. Rows represent the secondary structure classes and columns denote the amino acids of the target. Sum of the scores in each column is equal to 1.

The scores in a structural profile can be considered as soft labels and included into prediction models as soft constraints. Note that since structural profiles also use label information of template proteins they can be evaluated separately from the methods that use sequence profiles only. In another category, secondary structure information of proteins that are highly similar to target protein can be used directly for prediction as hard constraints. In this regard, using structural profiles can be considered as a category between the case where the sequential profiles are used only and the case where the label information from templates is used directly.

If the template proteins that are used to construct the structural profile are closer to target then structure can be predicted more accurately. Furthermore, in cases where the target protein resembles a sub-region instead of the entire protein (local similarity), one can expect improvement in the accuracy of secondary structure prediction at those local regions. The improvement will be more significant when the resembling region is longer.

To compute the HHMAKE PSSMs, the HMM-profile model of the target proteins in CB513 are aligned to the HMM-profile models of PDB proteins, which have true secondary structure labels available. In the next step, templates for which the percentage of sequence identity score is above 20% are eliminated in order to realize the single-sequence condition (allowing matches to distant templates only). Finally the frequency of occurrence counts are computed for the three secondary structure labels using labels of hit proteins that are aligned to amino acid positions of the target (represented by the columns of the profile matrix). This is followed by column normalization so that the sum of scores in each column is 1. If there is no match to an amino acid of the target,

then a value of 1/3 is assigned to all the profile values in the corresponding column. This can happen due to the fact that the alignments computed by HHBlits are local.

3.3 DSPRED METHOD

The DSPRED method is a two-stage classifier developed for the estimation of one dimensional structural properties such as secondary structure, dihedral angles and solvent accessibility. A similar approach has also been used for predicting dihedral angle classes replacing the support vector machine with neural networks [45, 46]. The steps of the DSPRED method is depicted in Figure 3.3.1. In DSPRED, separate dynamic Bayesian networks (DBNs) are trained for position specific scoring matrices obtained from PSI-BLAST [41] and HHBlits [42] methods. These input features are denoted as PSIBLAST PSSM and HHMAKE PSSM, respectively.

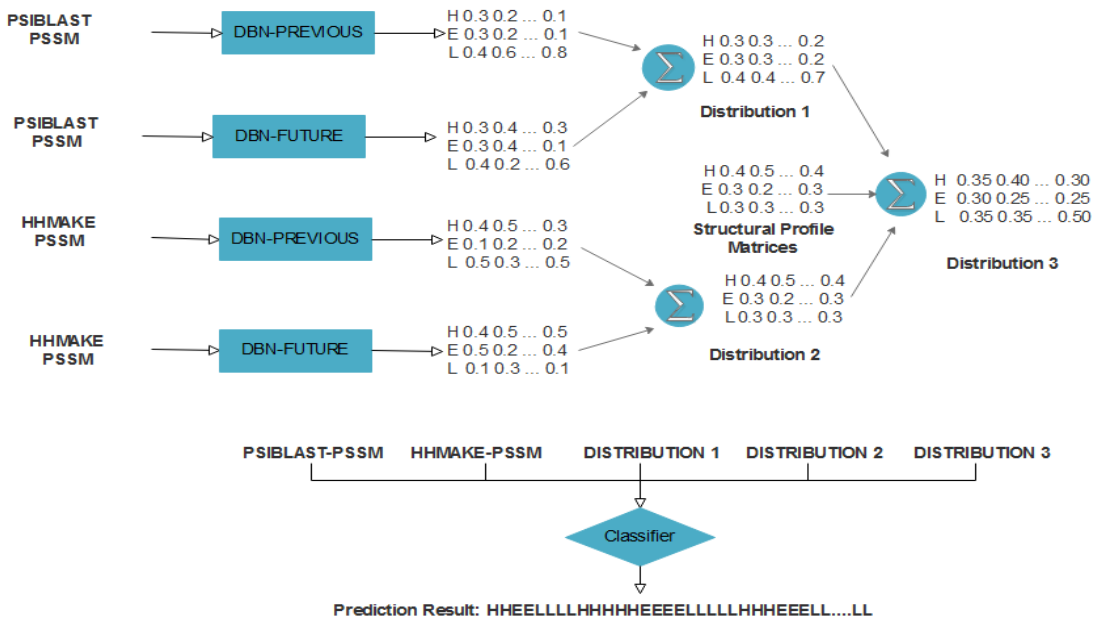
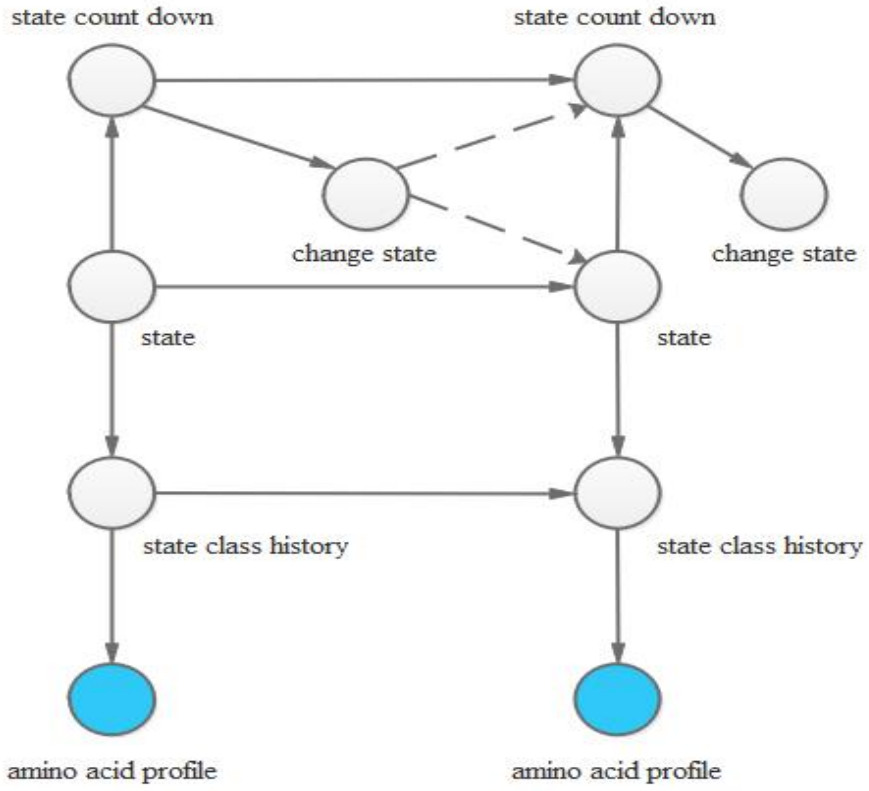


Figure 3.3.1 The two-stage hybrid model for estimating the 3-state secondary structure using dynamic Bayesian networks and a support vector machine

There are two types of DBN models in DSPRED. DBN-Previous represents the model in which the probability density of the profile vector at a given amino acid position depends on previous positions and DBN-future represents the model in which the probability density of the profile vector at a given position depends on subsequent positions. The profile vectors are the columns of the profile matrix and there are as many columns as the number of amino acids. The output of each DBN is a marginal a

posteriori probability distribution for the secondary structure class labels given the input features. These distributions are combined through averaging to obtain the predictions for each feature type. For example, Distribution 1 represents the average of the predictive distributions produced by DBNs that use PSI-BLAST PSSMs, Distribution 2 represents the average of the distributions generated by DBNs that use HHMAKE PSSM features and Distribution 3 is computed as the average of the Distribution 1, Distribution 2 and the structural profile matrix obtained using the HHBlits method. In this problem, since the number of secondary structure classes is three, the dimension of Distribution 1, 2, and 3 is 3×U where U is the number of amino acids. Consequently, each column contains the estimated probabilities of secondary structure classes at an amino acid position. In the second stage of DSPRED, the profile matrices (PSI-BLAST and HHMAKE) are combined with Distributions 1, 2, and 3 and sent to a discriminative classifier such as support vector machine. A symmetrical window is taken around each amino acid at which the secondary structure class is going to be predicted and features from these columns are concatenated to construct the input feature vector. The classifier in the second stage gives an estimate of the secondary structure label of the amino acid at the center of the window.



(A)

State	H	H	H	H	H	L	L	L	H	H	H	E	E	E	E	E	L	L	L	L	L	L
state count down	5	4	3	2	1	3	2	1	3	2	1	5	4	3	2	1	5	5	4	3	2	1
change state	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0

(B)

Figure 3.3.2 A) A dynamic Bayesian network for protein secondary structure prediction. B) The variables used for modeling the secondary structure segments. State variable represents the secondary structure class label. The state count down (with Dmax=5) shows the number of remaining amino acids from the current position until the next segment. Change state is used to signal transition from one segment to another.

The dynamic Bayesian network model used for predicting secondary structure class with 3 states is shown in Figure 3.3.2 [31]. Dynamic Bayesian network (DBN) is a generative model and is the superset of Hidden Markov Model (HMM) [47]. DBNs model the generation of profile vectors from hidden class variables obeying certain probability rules. The nodes of the DBN in Figure 4.1.2 represent random variables. The state variable represents the secondary structure class label of an amino acid. The amino acid profile variable contains the profile vector of an amino acid and is observed during training and testing. These vectors correspond to the columns of the PSSMs produced by PSI-BLAST and HHMAKE (i.e. the first step of HHBlits). The current and previous secondary structure labels are concatenated and stored in state class history variable, which is used to fit a different conditional Gaussian distribution to each possible value of state class history. This conditional distribution is the likelihood of observing the amino acid profile given the state class history variable and is responsible from the generation of the input features. The state count down contains a distance value from the current position till the next secondary structure segment. This variable helps to model the length distribution of the segments. If the number of amino acids from the current position till the next segment (denoted by N_A) is less than a threshold (called Dmax) then the state count down becomes equal to N_A otherwise it is set to Dmax. For positions in which the state count down is less than Dmax the length distribution is estimated using the maximum likelihood approach, which employs the frequency of occurrence counts. For the remaining positions a geometric distribution is fit to the length distribution. Change state variable signals transition from one segment to another.

After the DBN models are trained using proteins with known structure labels, the predictions that maximize the marginal a posteriori probability of class labels can be

computed by efficient algorithms. In our thesis, Linux based GMTK software package (Graphical Models Toolkit) is used to implement the DBN models [48]. The GMTK uses the EM algorithm for model training and in this thesis we used the junction tree algorithm for computing the predictions.

3.4 Support Vector Machines

Support vector machine is among the discriminative classifiers in machine learning. The main purpose of SVM is to identify a hyperplane which makes the most appropriate discrimination between two or more classes [49].

SVM can classify both linear and nonlinear datasets. Suppose there are two classes. One can draw an infinitely many planes that separate data samples of these classes. At this point, the aim of SVM is to find the hyperplane that maximizes the distance between the samples closest to the plane.

3.4.1 Linear Separation

In linear separation, data from different classes can be linearly separable from each other in multiple ways (Figure 3.4.1.1). The goal is to find the plane H_0 such that the distance from the closest data points (also called support vectors) to this plane is maximized. This distance is called the margin. If two hyperplanes (H_1, H_2) are drawn that pass through these closest data samples H_0 becomes at the center of these two planes, which is the optimum separation hyperplane.

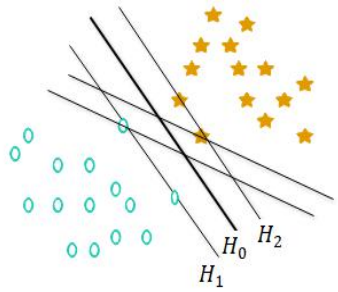


Figure 3.4.1.1 The linear SVM.

3.4.2 Non-linear Separation

In real word problems, many datasets are not linearly separable. In this case, the data cannot be split by a linear decision boundary, therefore a non-linear mapping is applied [50]. The data samples in input feature space are mapped to a higher dimensional space and a linear hyperplane is found that separates the data samples in new space. Formulating the problem in dual space and using the kernel trick the optimization problem can be solved without explicitly moving the data points to the

new space (though the kernel still defines a mapping to a non-linear feature space implicitly).

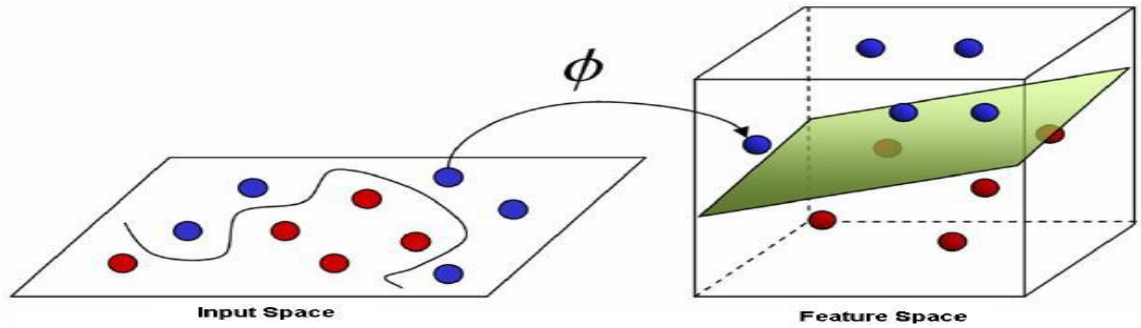


Figure 3.4.2.1 The non-linear SVM [51]

3.4.3 MULTICLASS SVM

The standard SVM is formulated for binary classification problem for separating two classes only. If there are three or more classes to be classified, a multi class SVM should be derived. Different approaches can be used to solve the multi class problem.

One versus one (OVO): In this approach, several binary classifiers are derived that separate pairs of classes and combined for the final prediction. For example, if there are three classes as 0, 1 and 2, a separate SVM is trained for 0 vs 1, 0 vs 2 and 1 vs 2. Then to classify a data sample a majority voting approach is used. This is summarized in the figure below.

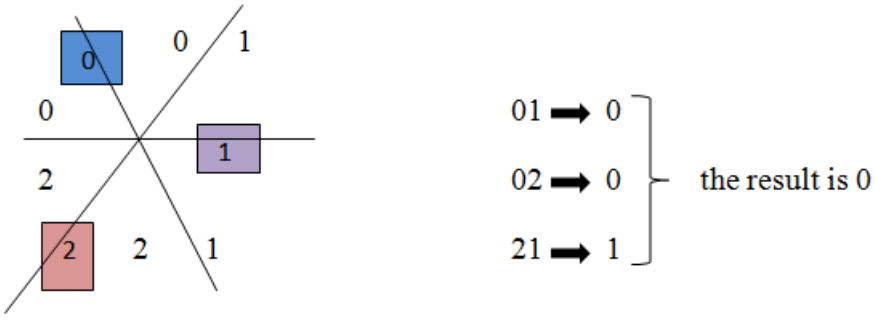


Figure 3.4.3.1 One-vs-one SVM

One versus all (OVA): In OVA technique, a binary classifier is trained that separates a class and the rest of the classes. In other words, for each classifier one class is considered as positive and all other classes as negative. Then, the final prediction can be obtained by choosing the SVM with maximum decision function output.

Multi Class Ranking: In this approach a single decision function is learned that aims to classify all classes. In this approach, classes of data samples are required to have an

ordering. A ranking function is learned that receives a feature vector as input and provides an output, which will be ranked according to the class ordering of the input. In other words, the ordering between the outputs of two data samples will be the same as the ordering between their classes. Ranking based SVM is the least preferred one because, the execution time can be high when compared to other approaches and a single function may not be found that can classify all the data samples.

3.5 RANDOM FOREST

Random forest is a method that forms an ensemble of decision trees used for classification or regression [52]. It is applied to many different problems including biomedical [53], physics, health, and bioinformatics. It is among the models that apply the bagging technique and combine the decisions of its base learners by weighted majority voting. Each tree is trained with a different subset of features selected randomly from the original feature set. Furthermore each tree model is constructed using a slightly different train set obtained by bootstrap sampling. To construct decision trees, Gini index is used as the impurity measure. In this thesis random forest is implemented using WEKA [54].

Random forests are preferred for a number of reasons. They are robust against overfitting. Increasing the number of features will not directly cause overfitting because a subset of features is randomly selected for each decision tree, which learns a specific concept in training data. As the dimension of the dataset increases the number of trees can be increased to learn all the concepts in training set. Therefore if the number of trees and the number of randomly selected features are selected properly a random forest can avoid the problem of overfitting.

The other advantage of random forest is its efficient performance on big data. Convenience of setting parameters, handling missing values are among other advantages of random forests.

Recently, random forests are compared with other classification models. For example, a comparison is made between random forest and SVM for microarray-based cancer classification, in which the SVM outperformed random forest [55]. Lee et al. [56] compared random forest with SVM to identify protein function using features that

are obtained from protein sequences attributes. They applied a correlation-based feature selection and compared them against the SVM and random forest models trained without feature selection. SVM with feature selection outperformed the random forest with and without feature selection.

3.6 Deep Convolutional Neural Fields

Deep learning is a machine learning technique that aims to learn highly non-linear and complex relationships in data. It has been applied to many fields such as image recognition [57], bioinformatics [58, 59], natural language processing [60, 61]. The goal of deep learning approaches is to learn high level and more complex attributes using lower level attributes that are simpler [62]. Learning attributes at multiple levels of abstraction provide an opportunity for a system for resolving complicated functions that maps the inputs to an output directly from data.

Deep learning has been improved in time. Firstly, deep belief network with Restricted Boltzmann Machines (RBM) [63] was introduced. This is followed by auto-encoders and other algorithms.

3.6.1 Deep Convolutional Neural Networks

Convolutional neural networks (CNNs) are successful deep learning architectures because of the successful training of the hierarchical layers. In CNN, the convolution has replaced the general matrix multiplication in standard neural networks (NN). This way and due to weight sharing the number of weights is decreased, thereby reducing the complexity of the network. Another advantage of CNN is its minimal pre-processing requirement because of the inherent feature extraction capability.

3.6.2 Deep Convolutional Neural Fields

Conditional neural fields are developed due to the inconvenience of resolving the nonlinear connection between input features and output layer of CRF, especially for sequence labeling [32]. A deep convolutional neural field (CNF) is a hybrid classifier obtained by combining a deep CNN [64] and a conditional random field (CRF) [32]. As a deep learning technique, it enables to capture highly non-linear relationships between

the input features and the output. Furthermore it can model the correlation between the contiguous output labels by the CRF model in the last layer. As a result, it combines the advantages of CNN and CRF trained jointly in a single model. A deep CNF architecture is shown in Figure 3.6.2.1.

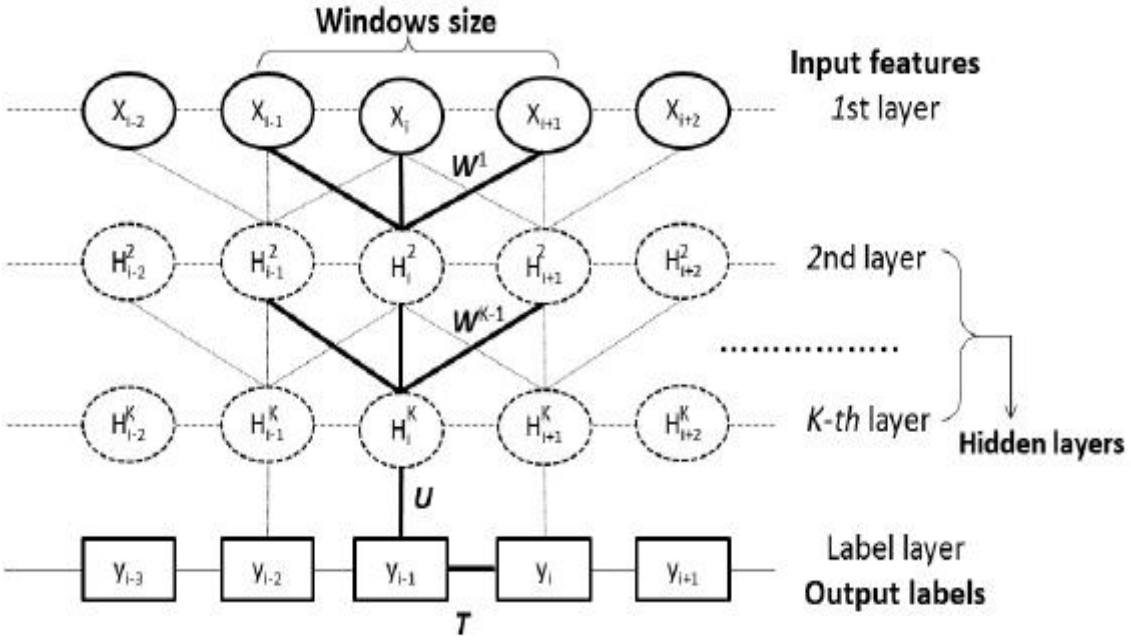


Figure 3.6.2.1 Design of Deep CNF [65], where *i* is the position index and X_i the associated input features, while H^k is for the *k-th* hidden layer and Y_i is for the output label. All the layers from the first to the top layer constitute a DCNN with parameter $W^k\{k = 1, 2, \dots, K\}$. The top layer and the label layer constitute a CRF with *U* and *T* as a model parameters. *U* determines association between output of the top layer and the label layer and *T* is used for adjacent label correlation.

3.7 Model Evaluation by Cross-Validation

Datasets used in machine learning, data mining generally split into two parts as train and test set to evaluate predictive models. Train set is used to train the model and test set to evaluate the accuracy. K-fold cross validation is a technique to test the prediction model by dividing the data set into k equal sized subsets. In each iteration, one of the k subsets is used as the test set and the other k-1 subsets form the training set. This procedure is repeated k times until all subsets are used as test set.

In this thesis, seven fold cross validation is used to evaluate the accuracy of the methods. For this purpose, the CB513 dataset is divided into seven equal sized folds by randomly assigning proteins into each subset. For each iteration of cross-validation a

subset is selected as test data and the remaining subsets are used as train data. Before evaluating the cross-validation accuracy, first the hyper-parameters of the models are optimized separately for each cross-validation iteration. For this purpose, 10% of the proteins in each train set are randomly selected as validation set. The remaining 90% of the train set and the validation set are used to optimize the hyper-parameters of the models. After the optimization process is finished, all proteins in the training set are used to learn the models and predictions are computed on the corresponding test set. This procedure is repeated for each fold and seven accuracy rates are obtained for each model. The overall accuracy is calculated by taking the average of these accuracy values.

3.8 Parameter Optimization

There are multiple conditions that may affect the performance of a classifier. One of these is the selection of the correct hyper-parameters, which cannot be directly learned from the regular training process. Optimizing these parameters enable us to fine-tune the model complexity and prevent over-fitting as well as under-fitting.

For optimization the classifier models are trained for various hyper-parameter combinations on a randomly selected subset that contains 90% of proteins from a train set (a total of 7 due to cross-validation) and predictions are computed on the corresponding validation set (randomly selected as 10% of train set). The hyper-parameters that maximize the overall accuracy on validation set are chosen as the optimum values. This procedure is repeated seven times for each iteration of the cross-validation experiment.

3.8.1 Parameter Optimization for Support Vector Machines

For SVM, we optimized C and Gamma (γ). C is a cost function parameter that controls the effect of each support vector. Selecting an appropriate value for C is important for tolerating the error. When the value of C is low, the decision is gets smoother and margin becomes larger. When the value of C is high, sensitivity at learning phase increases because of the decrease in margin.

Gamma parameter determines the amount of spreading influence. The decision boundary becomes closer to linear when the value of gamma parameter is small. When gamma gets higher value, the decision boundary becomes non-linear.

For SVM optimization a grid of C and gamma values are considered. For each parameter we selected the following values.

C = (2^{-5}, 2^{-3},2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}, 2^{13})

Gamma = (2^{-15}, 2^{-13},2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1},2^{3})

As a result, there are a total of 100 different combinations to consider. Once the optimum combination is found for a given cross-validation iteration (one that has the best accuracy on validation set) the model is trained on the full training set and predictions are computed on the test set. This procedure is repeated for other iterations of cross-validation experiment.

3.8.2 Parameter Optimization for Random Forest

The only parameter that is optimized for random forest is the number of trees (also called number of iterations in WEKA). The following values are considered for each fold of cross-validation.

Number of trees = (5 10 15 20 25 50 75 100 125 150 175 200 225 250 275 300 325 350 375 400 425 450 475 500).

3.8.3 Parameter Optimization for Deep Convolutional Neural Fields

The parameters that are optimized for deep CNF are the number of hidden layers, the number of hidden units in each layer, the width of the two dimensional kernel window applied at each hidden layer and the regularization coefficient. The window size is specified by a variable called window string. For example, when window string is 5, kernel window size will be 11 at each layer. The number of hidden nodes and number of hidden layers are specified by a variable called node string. For example if the node string is “50,50,50,50,50” the network contains five hidden layers with 50 hidden nodes in each layer. In deep CNF L_2-norm regularization is used. The following values are considered in the parameter grid used for optimization

Number of Hidden Layers = (3, 4, 5)

Number of Hidden Units = (75, 100, 125)

Kernel Window Size= (3, 4, 5)

Regularization Coefficient = (10, 50, 100)

3.9 Ensemble Methods

The basic idea behind ensemble methods is to combine the predictions of several classifiers for the purpose of improving prediction accuracy and robustness of the models. It is found that the performance of the final classifier can be improved by forming a method whose output is computed by combining the outputs of individual classifiers.

3.9.1 Model Averaging

Model averaging computes the weighted sum of prediction scores from multiple methods to generate a consensus prediction. Combining classifiers by model averaging generally achieves better results than any of the individual classifiers because of the possibility of reducing variance. The procedure is formulated as

$$p^c(y|\boldsymbol{X}) = \frac{1}{N}\sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} p^n(y|\boldsymbol{X}) \qquad \textbf{(3.9.1.1)}$$

where $\boldsymbol{X}$ is the input feature vector of an amino acid, y is the output class label (which can be H, E, or L representing helix, strand or loop, respectively), $p^c(y|\boldsymbol{X})$ is the posteriori probability of class label given feature vector, n is the index of the model used in the ensemble, $p^n(y|\boldsymbol{X})$ is the posteriori probability of class label given input vector from the n^{th} model, and N is the number of models in the ensemble. The final class prediction is obtained by selecting the secondary structure labels that maximize the $p^c(y|\boldsymbol{X})$. In this thesis, model averaging is applied to combine predictions obtained by SVM, deepCNF and random forest. The probability estimates are obtained for SVM using the –b option in libSVM [66] and for random forest using the –distribution option in WEKA [54]. Deep CNF automatically provides probability scores as the output [67]. The following combinations are considered in the ensemble

- SVM + random forest
- SVM + deep CNF
- deep CNF + random forest
- SVM + deep CNF + random forest

Chapter 4

4. Results

For SVM and random forest, our train and test datasets contain 49 features for each amino acid after concatenating 20 PSSM features for PSI-BLAST, 20 PSSM features for HHMAKE, 3 features for Distribution 1, 3 features for Distribution 2, 3 features for Distribution 3. When we repeat the concatenation for all amino acids within the window of size 11 around the center amino acid, we obtain a total of 539 features. In deepCNF we excluded Distributions 1 and 2 from the feature set and represented each amino acid by 43 features because in our preliminary tests this combination gave slightly better accuracy (result not shown). Due to applying a kernel window of size 11 in the temporal domain effectively deepCNF uses 11*43=473 features.

We train all models using the CB513 dataset and applied seven fold cross validation to determine the model hyper-parameters for each training method. . We divide our dataset into 7 equal size subsets for 7-fold cross validation. Each subset is used as test to validate and other 6 subsets are used as training. 10% of train test splits are taken as validation. These were randomly selected at the protein level. After selecting all each fold as test set and evaluate the model, 7 accuracy rates are obtained. Overall accuracy is calculated by taking average of 7 results obtained.

4.1 Optimization Results

The hyper-parameters of the models are optimized for each iteration of the seven fold cross-validation experiment on CB513. The following sections summarize the

optimum parameters found and the accuracy on validation sets for the SVM, random forest and deep CNF when models are trained using the optimums.

4.1.1 SVM Optimization

Table 4.1.1.1 shows the optimum C and gamma parameters for the SVM as well as the Q_3 accuracy on validation sets. A separate optimum is found for each fold of the cross-validation experiment on CB513. A tie occurred for the fourth fold.

Fold Number	**C parameter**	**Gamma parameter**	$\boldsymbol{Q_3}$
1	32	0.00195313	84.0
2	32	0.00195313	81.3
3	2	0.03125	84.3
4	32	0.00195313	83.7
	8192	0.000122070	83.7
5	2	0.0078125	84.5
6	2	0.03125	82.7
7	2048	0.000122070	83.2
Overall Accuracy:			**83.4**

Table 4.1.1.1 Optimum C and gamma parameters and overall accuracy of SVM on validation sets of CB513.

Table 4.1.1.2 shows the Q_3, Q_H, Q_E, Q_L measures for the SVM on validation sets of CB513. Q_3 is the overall accuracy and $\mathrm{Q_H}, \mathrm{Q_E}, \mathrm{Q_L}$ are the recall values for secondary structure classes. A separate SVM model is trained for each fold using the optimum C, gamma combination. According to these results, prediction accuracy of helices and loops are close to each other and higher than the accuracy of strands.

Fold Number	$\boldsymbol{Q_3}$	$\boldsymbol{Q_H}$	$\boldsymbol{Q_E}$	$\boldsymbol{Q_L}$
1	84.0	86.6	74.6	86.0

2	81.4	84.0	69.0	85.2
3	84.3	85.6	77.3	87.0
4	83.8	84.7	79.0	86.0
	83.8	87.0	80.3	84.5
5	84.5	85.3	73.2	85.5
6	82.7	83.2	77.0	86.3
7	83.2	86.6	74.6	86.0
Overall Accuracy:	**83.4**	**85.2**	**75.7**	**85.8**

Table 4.1.1.2 Recall values for each class type and overall accuracy of the SVM on validation sets of CB513.

4.1.2 Random Forest Optimization

Table 4.1.2.1 shows the optimum number of trees for random forest and the Q_3 accuracy on validation sets. A separate optimum is found for each fold of the cross-validation experiment on CB513.

Fold Number	**Number of trees**	$\boldsymbol{Q_3}$
1	375	83.0
2	500	79.9
3	425	82.8
4	500	82.5
5	225	82.8
6	300	80.9
7	100	81.9
Overall Accuracy:		**81.9**

Table 4.1.2.1 Optimum number of trees and overall accuracy of random forest on validation sets of CB513.

Table 4.1.2.2 shows the Q_3, Q_H, Q_E, Q_L measures for the random forest on validation sets of CB513. Q_3 is the overall accuracy and Q_H, Q_E, Q_L are the recall values for secondary structure classes. A separate random forest model is trained for each fold using the optimum number of trees.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	83.0	84.5	70.8	87.2
2	79.9	82.1	65.7	85.0
3	82.8	83.1	73.4	88.0
4	82.5	82.6	75.1	86.7
5	82.8	85.0	75.8	84.4
6	80.9	81.7	69.4	86.2
7	81.9	82.7	74.3	85.4
Overall Accuracy:	**81.9**	**83.1**	**72.0**	**86.1**

Table 4.1.2.2 Recall values for each class type and overall accuracy of random forest on validation sets of CB513.

4.1.3 Deep CNF Optimization

Two types of optimization experiments are performed. The first one fixes the number of hidden layers and optimizes the remaining hyper-parameters. For this approach the number of hidden layers is set to 3, 4 and 5 one at a time and for each of these values the optimization experiment is repeated. The second approach optimizes the number of hidden layers together with the other hyper-parameters.

4.1.3.1 Deep CNF Optimization for Three Hidden Layers

Table 4.1.3.1.1 shows the optimum hyper-parameters for deep CNF with 3 hidden layers and the Q_3 accuracy on validation sets. A separate optimum is found for each fold of the cross-validation experiment on CB513. The optimum regularization coefficient is

obtained as 50, the optimum number of hidden nodes is obtained as 100 and the optimum kernel size is obtained as 3-4 in most of the cross-validation folds.

Number of Hidden Layers	**Fold Number**	**Window string**	**Node string**	**Reg. Coefficient**	Q_3
3	1	3,3,3	100,100,100	50	90.5
3	2	3,3,3	100,100,100	10	88.8
3	3	3,3,3	125,125,125	50	92.0
3	4	4,4,4	100,100,100	50	88.9
3	5	4,4,4	125,125,125	50	89.7
3	6	4,4,4	100,100,100	50	91.3
3	7	4,4,4	100,100,100	100	89.6
Overall Accuracy:					**90.1**

Table 4.1.3.1.1 Optimum kernel width (window string), number of hidden nodes (node string), regularization parameter and overall accuracy of deep CNF with three hidden layers on validation sets of CB513.

Table 4.1.3.1.2 shows the Q_3, Q_H, Q_E, Q_L measures for the deep CNF with three hidden layers on validation sets of CB513. Though the accuracy values are larger they are obtained on validation sets as part of the optimization process and could be due to overfitting. Therefore the performance evaluations should be based on predictions on test data.

Number of Hidden Layers	**Fold Number**	Q_3	Q_H	Q_E	Q_L
3	1	90.5	92.2	89.4	89.7
3	2	88.8	91.8	85.8	88.2
3	3	92.0	95.2	91.2	89.6
3	4	89.0	91.8	87.3	88.0
3	5	89.7	91.1	87.0	89.9
3	6	91.2	93.7	88.5	90.1

3	7	89.6	90.3	83.3	90.1
Overall Accuracy:		**90.1**	**92.3**	**87.5**	**89.3**

Table 4.1.3.1.2 Recall values for each class type and overall accuracy of deep CNF with three hidden layers on validation sets of CB513.

4.1.3.2 Deep CNF Optimization for Four Hidden Layers

Table 4.1.3.2.1 shows the optimum hyper-parameters for deep CNF with four hidden layers and the Q_3 accuracy on validation sets. The optimum regularization coefficient is obtained as 10 or 50, the optimum number of hidden nodes is obtained as 75, 100 or 125 and the optimum kernel size is obtained as 3 or 4.

Number of Hidden Layers	**Fold Number**	**Window string**	**Node string**	**Reg. Coefficient**	Q_3
4	1	3,3,3,3	75,75,75,75	100	90.6
4	2	3,3,3,3	100,100,100,100	10	88.8
4	3	4,4,4,4	125,125,125,125	50	91.9
4	4	3,3,3,3	125,125,125,125	50	88.9
4	5	3,3,3,3	75,75,75,75	10	89.8
4	6	3,3,3,3	125,125,125,125	50	91.3
4	7	3,3,3,3	75,75,75,75	50	89.6
Overall Accuracy:					**90.2**

Table 4.1.3.2.1 Optimum kernel width (window string), number of hidden nodes (node string), regularization parameter and overall accuracy of deep CNF with four hidden layers on validation sets of CB513.

Table 4.1.3.2.2 shows the Q_3, Q_H, Q_E, Q_L measures for the deep CNF with four hidden layers on validation sets of CB513. Similar accuracy values are obtained as in the model with three hidden layers.

Number of Hidden Layers	**Fold Number**	Q_3	Q_H	Q_E	Q_L
4	1	90.4	92.4	88.5	89.5

4	2	88.5	90.9	85.2	88.8
4	3	91.8	94.7	91.2	89.6
4	4	89.0	92.2	87.6	87.8
4	5	89.6	90.6	87.9	89.6
4	6	91.4	93.8	89.5	90.4
4	7	89.4	89.6	83.8	91.8
Overall Accuracy:		**90.0**	**91.9**	**87.7**	**89.6**

Table 4.1.3.2.2 Recall values for each class type and overall accuracy of deep CNF with four hidden layers on validation sets of CB513.

4.1.3.3 Deep CNF Model with 5-Hidden Layers

Table 4.1.3.3.1 shows the optimum hyper-parameters for deep CNF with five hidden layers and the Q_3 accuracy on validation sets. The optimum regularization coefficient is obtained as 50, the optimum number of hidden nodes is obtained as 75 or 125 and the optimum kernel size is obtained as 3 or 4.

Number of hidden layers	**Fold Number**	**Window string**	**Node string**	**Reg. Coefficient**	Q_3
5	1	3,3,3,3,3	125,125,125,125,125	50	90.7
5	2	4,4,4,4,4	125,125,125,125,125	50	88.7
5	3	4,4,4,4,4	75,75,75,75,75	50	91.8
5	4	4,4,4,4,4	125,125,125,125,125	50	89.1
5	5	3,3,3,3,3	125,125,125,125,125	50	89.8
5	6	3,3,3,3,3	75,75,75,75,75	50	91.4
5	7	3,3,3,3,3	125,125,125,125,125	50	89.5
				100	
Overall Accuracy:					**90.1**

Table 4.1.3.3.1 Optimum kernel width (window string), number of hidden nodes (node string), regularization parameter and overall accuracy of deep CNF with five hidden layers on validation sets of CB513.

Table 4.1.3.3.2 shows the Q_3, Q_H, Q_E, Q_L measures for the deep CNF with five hidden layers on validation sets of CB513. Similar accuracy values are obtained as in the model with three and four hidden layers.

Number of Hidden Layers	**Fold Number**	Q_3	Q_H	Q_E	Q_L
5	1	90.4	92.2	89.4	89.0
5	2	88.6	91.1	86.1	88.3
5	3	91.6	94.8	91.7	89.0
5	4	88.8	90.8	86.5	88.9
5	5	89.6	91.5	89.0	88.4
5	6	91.1	94.0	89.5	89.5
5	7	89.4 **90.4**	**90.2**	**85.2**	**90.9**
Overall Accuracy:		**90.0**	**91.9**	**87.7**	**89.6**

Table 4.1.3.3.2 Recall values for each class type and overall accuracy of deep CNF with five hidden layers on validation sets of CB513.

4.1.3.4 Deep CNF Optimization for all Hyper-parameters

In this part, the number of hidden layers is also optimized together with the number of hidden nodes, kernel width, and regularization coefficient. Table 4.1.3.4.1 shows the optimum hyper-parameters for deep CNF and the accuracy values on validation sets. The optimum regularization coefficient is obtained as 10 or 50, the optimum number of hidden nodes is obtained as 75, 100 or 125 and the optimum kernel size is obtained as 3 or 4.

Fold Number	**Number of hidden layers**	**Window string**	**Node string**	**Reg. Coefficient**	Q_3
1	5	3,3,3,3,3	125,125,125,125,125	50	90.7

2	4	3,3,3,3	100,100,100,100	10	88.8
	3	3,3,3	100,100,100	10	88.8
3	3	3,3,3	125,125,125	50	92.0
4	5	4,4,4,4,4	125,125,125,125,125	50	89.0
5	5	3,3,3,3,3	125,125,125,125,125	50	89.8
6	5	3,3,3,3,3	75,75,75,75,75	50	91.4
7	3	4,4,4	100,100,100	100	89.6
Overall Accuracy:					**90.2**

Table 4.1.3.4.1 Optimum kernel width (window string), number of hidden layers, number of hidden nodes (node string), regularization parameter and overall accuracy of deep CNF with optimum number of hidden layers combination on validation sets of CB513.

Fold Number	**Number of hidden layers**	Q_3	Q_H	Q_E	Q_L
1	5	90.7	90.2	89.4	89.0
2	4	88.8	91.0	85.0	88.8
	3	88.8	91.8	85.0	88.0
3	3	92.0	95.0	91.2	89.0
4	5	89.0	90.0	86.5	89.0
5	5	89.8	91.5	89.0	88.4
6	5	91.4	94.0	89.5	89.5
7	3	89.6	90.3	83.3	91.2
Overall Accuracy:		**90.2**	**91.8**	**87.7**	**89.3**

Table 4.1.3.4.2 Recall values for each class type and overall accuracy of deep CNF with optimum number of hidden layers on validation sets of CB513.

Table 4.1.3.4.2 shows the Q_3, Q_H, Q_E, Q_L measures for the deep CNF with hidden layers combination on validation sets of CB513. Similar accuracy values are obtained as in the model with three four and five hidden layers. This demonstrates that it is

sufficient to have three to five hidden layers for secondary structure prediction and the validation accuracies of the models for each hidden layer configuration are comparable to each other. Therefore there is not much gain when the number of hidden layers is greater than three.

4.2 Train-Test Results

Once the optimum hyper-parameters are found for each cross-validation iteration, models are trained on the full train sets and predictions are computed on test sets. Table 4.2.1 shows the recall and precision measures for each class type and the overall accuracy of the SVM on test sets of CB513. As shown in the table, accuracies are approximately between 82% and 83% on test data, except for the seventh fold, which has 85% accuracy. Q_E is the recall measure for strands secondary and is the lower than the accuracy of other class types.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.1	85.3	72.0	85.7	89.6	84.7	75.6
2	82.2	85.0	76.5	83.4	86.0	84.2	78.6
3	82.9	84.6	75.8	85.5	88.7	84.9	77.9
4	82.7	84.1	75.7	85.0	88.3	82.5	78.7
5	82.2	84.1	78.1	82.7	88.7	78.3	79.3
6	82.5	84.9	76.7	83.5	88.5	79.5	79.7
7	85.0	87.3	78.2	85.8	91.4	84.2	79.5
Overall Accuracy:	**82.8**	**85.1**	**76.1**	**84.5**	**88.9**	**82.6**	**78.5**

Table 4.2.1 Recall and precision measures for each class type and overall accuracy of SVM on test sets of 7-fold cross-validation on CB513.

Table 4.2.2 shows the recall and precision measures for each class type and the overall accuracy of the random forest on test sets of CB513. As shown in this table,

generally, accuracies obtained for each fold have close rates. Overall accuracy is 81.8%, which is 1% lower than the accuracy of SVM.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	81.1	83.2	70.5	86.0	89.8	84.2	73.9
2	81.4	82.4	75.2	84.3	87.4	82.6	77.3
3	81.8	82.2	72.7	86.4	89.9	83.9	75.5
4	81.5	82.8	73.3	84.8	87.6	81.8	77.3
5	81.2	81.9	76.1	83.2	88.7	78.5	77.3
6	82.0	82.8	74.5	85.1	89.8	79.2	78.2
7	83.4	86.0	73.7	85.4	90.5	83.6	77.2
Overall Accuracy:	**81.8**	**83.2**	**73.7**	**85.0**	**89.2**	**81.9**	**76.7**

Table 4.2.2 Recall and precision measures for each class type and overall accuracy of random forest on test sets of 7-fold cross-validation on CB513.

Table 4.2.3 shows the recall and precision measures for each class type and the overall accuracy of the deep CNF with three hidden layers on test sets of CB513. The overall accuracy is obtained as 82.6%, which is comparable to the accuracy of the SVM.

Fold Number	**Hidden Layer Number**	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	3	81.9	86.7	73.5	83.1	88.6	82.5	76.6
2	3	81.6	86.0	76.5	81.4	84.4	83.2	78.8
3	3	82.8	84.7	75.4	85.4	88.5	84.6	78.0
4	3	82.4	86.1	76.6	82.6	86.6	80.8	80.0
5	3	82.0	85.4	78.3	81.2	87.7	77.5	80.0
6	3	82.5	85.3	78.3	82.5	88.4	78.1	80.4
7	3	84.5	87.7	77.1	84.9	90.6	83.6	79.4
Overall Accuracy:		**82.6**	**86.0**	**76.5**	**83.0**	**88.0**	**81.4**	**79.1**

Table 4.2.3 Recall and precision measures for each class type and overall accuracy of deep CNF with three hidden layers on test sets of 7-fold cross-validation on CB513.

Table 4.2.4 shows the recall and precision measures for each class type and the overall accuracy of the deep CNF with four hidden layers on test sets of CB513. The overall accuracy is obtained as 82.6%, which is comparable to the accuracy of the SVM and the deep CNF with three hidden layers.

Fold Number	**Hidden Layer Number**	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	4	82.1	85.2	73.5	84.9	90.3	83.0	75.9
2	4	82.0	85.5	76.5	82.8	85.3	84.1	78.8
3	4	82.7	85.4	76.0	84.1	87.2	84.8	78.3
4	4	82.4	85.9	76.5	82.8	86.5	81.3	79.9
5	4	81.8	84.8	77.7	81.6	87.3	78.8	79.3
6	4	82.4	84.6	77.1	83.4	88.5	79.1	79.8
7	4	84.6	88.2	78.2	84.1	90.3	83.1	79.9
Overall Accuracy:		**82.6**	**85.8**	**76.5**	**83.3**	**88.0**	**82.0**	**78.9**

Table 4.2.4 Recall and precision measures for each class type and overall accuracy of deep CNF with four hidden layers on test sets of 7-fold cross-validation on CB513.

Table 4.2.5 shows the recall and precision measures for each class type and the overall accuracy of the deep CNF with five hidden layers on test sets of CB513. The overall accuracy is obtained as 82.6%, which is comparable to the accuracy of the SVM and the deep CNFs with three and four hidden layers.

Fold Number	**Hidden Layer Numbers**	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	5	82.1	86.2	73.1	84.4	89.6	83.5	76.1
2	5	81.9	86.6	77.0	81.4	84.4	83.0	79.6

3	5	82.9	85.7	75.6	84.5	87.7	84.8	78.3
4	5	82.0	86.8	76.3	81.3	85.3	80.3	80.3
5	5	81.9	85.5	77.7	81.3	87.3	77.9	79.9
6	5	82.7	85.6	77.0	83.2	88.1	79.2	80.4
7	5	84.6	88.3	77.4	84.3	90.2	83.6	79.8
Overall Accuracy:		**82.6**	**86.5**	**76.3**	**82.9**	**87.6**	**81.7**	**79.2**

Table 4.2.5 Recall and precision measures for each class type and overall accuracy of deep CNF with five hidden layers on test sets of 7-fold cross-validation on CB513.

Table 4.2.6 shows the recall and precision measures for each class type and the overall accuracy of the deep CNF with optimum number of hidden layers on test sets of CB513. The optimum values are also used for the other hyper-parameters. The overall accuracy is obtained as 82.6%, which is comparable to the accuracy of the SVM and the deep CNFs with three four and five hidden layers.

Fold Number	**Number of hidden layer**	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	5	82.1	86.2	73.1	84.4	89.6	83.5	76.1
2	4	82.0	85.5	76.5	82.8	85.3	84.1	78.8
	3	81.6	86.0	76.5	81.4	84.4	83.2	78.8
3	3	82.8	84.7	75.4	85.4	88.5	84.6	78.0
4	5	82.0	86.8	76.3	81.3	85.3	80.3	80.3
5	5	81.9	85.5	77.7	81.3	87.3	77.9	79.9
6	5	82.7	85.7	77.0	83.2	88.1	79.2	80.4
7	3	84.5	87.7	77.2	85.0	90.6	83.6	79.4

Overall Accuracy:	**82.6**	**86.1**	**76.2**	**83.3**	**87.9**	**81.8**	**79.0**

Table 4.2.6 Recall and precision measures for each class type and overall accuracy of deep CNF with optimum number of hidden layers on test sets of 7-fold cross-validation on CB513.

4.3 Ensemble Method Results

4.3.1 Model Averaging Results on Validation and Test Sets

4.3.1.1 Support Vector Machines and Random Forest

In the first ensemble, we applied model averaging to combine the outputs of support vector machine and random forest. For this purpose, predicted probability scores are obtained for the three classes and for each amino acid. Then, the average of these scores are computed and the class label with the maximum score is selected as the prediction. Tables 4.3.1.1.1 and 4.3.1.1.2 show the results of combining support vector machine and random forest on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 83.2% on validation sets and 82.8% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	84.0	85.8	73.8	87.5
2	83.0	83.8	67.8	85.6
3	83.0	85.0	76.0	87.8
4	83.0	83.7	77.5	86.7
5	83.0	86.2	78.4	85.0
6	83.0	84.2	71.4	86.0
7	83.0	83.4	76.9	86.7
Overall Accuracy:	**83.2**	**84.6**	**74.5**	**86.5**

Table 4.3.1.1.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM and random forest on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	PrecisionH	PrecisionE	PrecisionL
1	81.9	84.4	71.3	86.4	90.1	85.0	74.9
2	82.5	84.5	76.5	84.5	87.2	84.3	78.6
3	82.8	83.9	74.9	86.3	89.3	85.2	77.3
4	82.5	83.7	75.2	85.4	88.4	82.8	78.5
5	82.0	83.2	77.6	83.2	88.9	78.9	78.7
6	82.7	84.3	76.0	84.8	89.4	80.2	79.3
7	84.7	86.9	76.6	86.3	91.2	85.0	78.8
Overall Accuracy:	**82.8**	**84.5**	**75.4**	**85.2**	**89.3**	**83.0**	**78.1**

Table 4.3.1.1.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM and random forest on test sets of 7-fold cross-validation on CB513.

4.3.1.2 Support Vector Machines and Deep CNF

As a second ensemble, model averaging is applied to support vector machines and deep convolutional neural field method. Here, we used the deep CNF that has the optimum number of hidden layers. Tables 4.3.1.2.1 and 4.3.1.2.2 show the results of combining support vector machine and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 83.5% on validation sets and 83.0% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	84.0	87.0	75.2	86.6
2	83.0	85.3	70.8	84.3
3	83.0	85.3	77.6	87.0
4	84.0	85.5	79.8	85.5
5	84.0	87.5	80.3	84.0
6	83.0	85.4	73.2	85.0

7	83.0	83.8	77.5	85.6
Overall Accuracy:	**83.5**	**85.7**	**76.3**	**85.4**

Table 4.3.1.2.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with optimum number of hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.5	86.0	73.2	85.5	90.0	84.5	76.1
2	82.4	85.4	76.8	83.5	86.0	84.4	79.0
3	83.3	85.0	76.0	86.0	88.9	85.5	78.2
4	82.6	85.7	76.5	83.3	86.8	81.8	79.8
5	82.2	84.6	78.0	82.4	88.4	78.4	79.6
6	82.9	85.4	77.2	83.9	88.7	79.8	80.3
7	84.9	87.5	77.8	85.6	91.1	84.1	79.6
Overall Accuracy:	**83.0**	**85.7**	**76.5**	**84.3**	**88.7**	**82.6**	**79.0**

Table 4.3.1.2.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with optimum number of hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.3 Support Vector Machines and Deep CNF with Three Hidden Layers

The third ensemble combines support vector machine and deep convolutional neural field with three hidden layers. Tables 4.3.1.3.1 and 4.3.1.3.2 show the results of combining support vector machine and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 83.4% on validation sets and 83.0% on test sets. These are similar to the results obtained when the deep CNF with optimum number of hidden layers is used.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	84.0	87.0	75.7	86.0

2	83.0	85.0	70.8	84.3
3	83.0	85.7	77.0	86.8
4	83.0	84.6	79.8	85.5
5	84.0	87.5	80.3	84.0
6	83.0	85.3	71.7	85.6
7	83.0	84.0	77.5	85.7
Overall Accuracy:	**83.4**	**85.6**	**76.0**	**85.4**

Table 4.3.1.3.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with three hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.3	86.1	73.3	84.8	89.5	84.0	76.3
2	82.3	85.6	77.0	83.2	85.7	84.4	79.0
3	83.3	85.0	76.0	86.0	88.9	85.5	78.2
4	82.9	85.6	76.5	84.2	87.9	81.9	79.8
5	82.3	84.9	78.3	82.3	88.7	78.2	79.6
6	82.9	85.3	77.8	83.5	88.8	79.1	80.4
7	84.9	87.6	77.8	85.6	91.1	84.1	79.6
Overall Accuracy:	**83.0**	**85.8**	**76.6**	**84.2**	**88.8**	**82.4**	**79.1**

Table 4.3.1.3.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with three hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.4 Support Vector Machines and Deep CNF with Four Hidden Layers

The fourth ensemble combines support vector machine and deep convolutional neural fields with four hidden layers. Tables 4.3.1.4.1 and 4.3.1.4.2 show the results of combining support vector machine and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 83.3% on validation sets and 83.0% on test sets. These are similar to the

results obtained when the deep CNF with three or optimum number of hidden layers is used.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	84.0	87.0	75.2	86.6
2	83.0	84.5	69.4	85.3
3	83.0	85.3	77.6	87.0
4	83.0	85.5	79.8	85.5
5	84.0	87.0	80.7	84.0
6	83.0	85.0	73.0	85.0
7	83.0	83.7	77.7	85.7
Overall Accuracy:	**83.3**	**85.4**	**76.2**	**85.6**

Table 4.3.1.4.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with four hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.3	85.4	73.0	85.5	89.8	84.5	75.9
2	82.4	85.4	76.8	83.5	86.0	84.4	79.0
3	83.2	85.1	76.2	85.4	88.5	85.4	78.3
4	82.9	85.3	76.2	84.4	87.7	82.4	79.6
5	82.2	84.5	77.8	82.7	88.4	78.9	79.3
6	82.9	85.0	77.4	84.0	89.0	79.8	80.0
7	85.1	88.0	78.4	85.4	91.1	84.4	79.9
Overall Accuracy:	**83.0**	**85.6**	**76.5**	**84.4**	**88.8**	**82.8**	**78.9**

Table 4.3.1.4.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with four hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.5 Support Vector Machines and Deep CNF with 5 Hidden Layer

The fifth ensemble combines support vector machine and deep convolutional neural field with five hidden layers. Tables 4.3.1.5.1 and 4.3.1.5.2 show the results of combining support vector machine and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 83.3% on validation sets and 83.0% on test sets. These are similar to the results obtained when the deep CNF with three, four or optimum number of hidden layers is used.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	84.0	87.0	75.3	85.8
2	83.0	84.5	69.8	85.3
3	83.0	85.4	78.0	87.0
4	83.0	85.2	79.8	85.0
5	84.0	88.0	80.3	83.5
6	83.0	85.5	73.2	85.0
7	83.0	83.8	77.5	85.6
Overall Accuracy:	**83.3**	**85.6**	**76.3**	**85.3**

Table 4.3.1.5.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with five hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.5	85.9	73.2	85.5	90.0	84.5	76.1
2	82.4	85.8	77.2	83.0	85.7	84.0	79.4
3	83.4	85.3	76.3	85.7	88.7	85.6	78.5
4	82.6	85.7	76.5	83.3	86.8	81.8	79.8
5	82.2	84.7	78.0	82.4	88.4	78.4	79.6

6	83.0	85.4	77.2	83.9	88.7	79.8	80.3
7	84.8	87.8	78.1	85.0	90.7	84.4	79.6
Overall Accuracy:	**83.0**	**85.9**	**76.6**	**84.1**	**88.5**	**82.6**	**79.1**

Table 4.3.1.5.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM and deep CNF with five hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.6 Random Forest and Deep Convolutional Neural Field

The sixth ensemble combines random forest and deep convolutional neural field with optimum number of hidden layers. Tables 4.3.1.6.1 and 4.3.1.6.2 show the results of combining random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 83.1% on validation sets and 82.8% on test sets. These are close to the results obtained for the other ensembles.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	84.0	86.0	73.0	87.0
2	83.0	85.7	69.5	84.4
3	83.0	84.7	75.6	87.8
4	83.0	84.7	78.2	86.0
5	83.0	86.5	78.8	84.4
6	83.0	84.0	71.5	85.3
7	83.0	83.7	76.0	83.0
Overall Accuracy:	**83.1**	**85.0**	**74.6**	**85.8**

Table 4.3.1.6.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with optimum number of hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.2	85.3	72.5	85.6	90.2	84.4	75.6

2	82.2	84.4	76.3	84.0	86.8	83.7	78.4
3	83.0	84.5	75.1	86.1	89.2	84.9	77.7
4	82.4	85.5	75.4	83.6	87.0	81.6	79.4
5	81.9	84.1	77.4	82.6	88.1	78.7	79.1
6	82.8	84.9	75.8	84.6	89.4	79.6	79.7
7	84.5	87.4	75.8	85.8	91.1	84.2	78.9
Overall Accuracy:	**82.8**	**85.2**	**75.5**	**84.6**	**88.9**	**82.4**	**78.5**

Table 4.3.1.6.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with optimum number of hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.7 Random Forest and Deep CNF with Three Hidden Layers

The seventh ensemble combines random forest and deep convolutional neural field with three hidden layers. Tables 4.3.1.7.1 and 4.3.1.7.2 show the results of combining random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.1% on validation sets and 82.7% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	82.0	86.3	73.0	86.0
2	82.0	85.7	69.5	84.0
3	82.0	85.0	75.5	87.6
4	82.0	84.0	78.2	86.0
5	82.0	86.5	78.8	84.4
6	82.0	83.6	70.0	85.7
7	83.0	84.0	76.0	85.4
Overall Accuracy:	**82.1**	**85.0**	**74.4**	**85.6**

Table 4.3.1.7.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with three hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	PrecisionH	PrecisionE	PrecisionL
1	82.0	85.5	72.8	84.7	89.6	83.3	75.8
2	82.2	84.8	76.3	83.6	86.4	83.7	78.6
3	83.0	84.5	75.1	86.1	89.2	84.9	77.7
4	82.6	85.0	75.8	84.2	87.7	81.5	79.4
5	82.0	84.1	77.6	82.7	88.8	78.4	79.0
6	82.7	84.5	76.9	84.1	89.2	79.2	79.8
7	84.5	87.4	75.8	85.8	91.1	84.2	78.9
Overall Accuracy:	**82.7**	**85.2**	**75.8**	**84.4**	**89.0**	**82.1**	**78.5**

Table 4.3.1.7.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with three hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.8 Random Forest and Deep CNF with Four Hidden Layers

The eighth ensemble combines random forest and deep convolutional neural field with four hidden layers. Tables 4.3.1.8.1 and 4.3.1.8.2 show the results of combining random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.1% on validation sets and 82.8% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	82.0	86.0	73.0	87.0
2	82.0	84.4	67.7	85.2
3	82.0	84.7	75.6	87.8
4	82.0	84.7	78.2	86.0
5	82.0	86.2	79.3	84.4

6	82.0	83.8	72.0	85.2
7	83.0	83.7	76.0	85.8
Overall Accuracy:	**82.1**	**84.8**	**74.5**	**86.0**

Table 4.3.1.8.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with four hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.1	84.8	72.4	86.0	90.6	84.2	75.4
2	82.2	84.4	76.3	84.0	86.8	83.7	78.4
3	82.9	84.5	75.3	85.7	88.5	85.4	77.8
4	82.6	85.1	75.4	84.3	87.9	81.7	79.2
5	81.9	83.8	77.3	82.7	88.2	79.2	78.7
6	82.8	84.3	76.5	84.8	89.4	80.1	79.6
7	84.6	87.8	76.4	85.4	90.9	84.4	79.1
Overall Accuracy:	**82.8**	**85.1**	**75.7**	**84.7**	**89.0**	**82.6**	**78.4**

Table 4.3.1.8.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with four hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.9 Random Forest and Deep CNF with 5 Hidden Layers

The ninth ensemble method combines random forest and deep convolutional neural field with five hidden layers classifiers. Tables 4.3.1.9.1 and 4.3.1.9.2 show the results of combining random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.5% on validation sets and 82.7% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L

1	82.2	86.0	74.0	86.3
2	82.3	84.2	68.0	85.3
3	82.2	84.8	76.2	87.7
4	82.2	84.8	77.5	85.0
5	82.3	87.0	79.2	84.2
6	82.3	84.0	71.5	85.3
7	83.3	83.7	760	85.8
Overall Accuracy:	**82.5**	**85.0**	**74.6**	**85.6**

Table 4.3.1.9.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with five hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.2	85.3	72.5	85.6	90.2	84.4	75.6
2	82.1	85.3	76.2	83.2	86.0	83.4	78.8
3	83.0	84.7	75.1	86.0	89.0	85.2	77.8
4	82.4	85.5	75.4	83.6	87.0	81.6	79.4
5	81.9	84.1	77.4	82.5	88.1	78.7	79.1
6	82.8	84.9	75.8	84.6	89.4	79.6	79.7
7	84.3	87.5	76.1	85.1	90.6	83.9	78.8
Overall Accuracy:	**82.7**	**85.4**	**75.5**	**84.3**	**88.7**	**82.4**	**78.5**

Table 4.3.1.9.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines random forest and deep CNF with five hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.10 Support Vector Machines, Random Forest and Deep CNF

The tenth ensemble combines support vector machines, random forest and deep CNF with optimum number of hidden layers. Tables 4.3.1.10.1 and 4.3.1.10.2 show the results of combining SVM, random forest and deep CNF on validation and test sets of

seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.6% on validation sets and 83.0% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	82.0	86.3	74.2	87.0
2	82.0	85.0	69.7	85.4
3	83.0	85.2	76.6	87.7
4	83.0	84.6	78.2	86.0
5	82.0	86.8	79.3	84.6
6	83.0	84.2	72.2	85.5
7	83.0	83.7	77.0	86.2
Overall Accuracy:	**82.6**	**85.1**	**75.3**	**86.0**

Table 4.3.1.10.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with optimum number of hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.4	85.5	72.5	86.1	90.4	84.8	75.8
2	82.5	85.0	76.6	84.1	86.8	84.4	78.8
3	83.1	84.5	75.1	86.4	89.3	85.4	77.8
4	82.7	85.1	76.0	84.4	87.5	82.3	79.5
5	82.0	84.0	77.7	82.5	88.6	78.2	79.0
6	83.0	85.0	76.7	84.5	89.3	79.9	79.9
7	84.8	87.4	77.0	86.1	91.3	84.8	79.2
Overall Accuracy:	**83.0**	**85.3**	**76.0**	**84.8**	**89.1**	**82.7**	**78.6**

Table 4.3.1.10.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with optimum number of hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.11 Support Vector Machines, Random Forest and Deep CNF with Three Hidden Layers

The eleventh ensemble combines support vector machine, random forest and deep convolutional neural field with three hidden layers. Tables 4.3.1.11.1 and 4.3.1.11.2 show the results of combining SVM, random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.4% on validation sets and 83.0% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	82.0	86.3	74.0	86.6
2	82.0	85.0	69.7	85.4
3	83.0	85.2	76.4	87.4
4	83.0	84.3	78.3	86.0
5	82.0	86.8	79.3	84.6
6	82.0	84.6	71.2	86.2
7	83.0	84.0	77.0	86.2
Overall Accuracy:	**82.4**	**85.2**	**75.1**	**86.0**

Table 4.3.1.11.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with three hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.2	85.5	72.7	85.6	90.1	84.3	75.7
2	82.4	85.1	76.5	84.0	86.6	84.2	78.8
3	83.1	84.5	75.1	86.4	89.3	85.4	77.8

4	82.8	85.0	75.6	84.8	88.1	82.4	79.2
5	82.1	84.0	77.8	83.0	89.0	78.4	79.1
6	82.9	85.0	77.1	84.3	89.2	79.6	80.1
7	84.8	87.4	77.0	86.1	91.3	84.8	79.2
Overall Accuracy:	**83.0**	**85.3**	**76.0**	**84.8**	**89.2**	**82.7**	**78.6**

Table 4.3.1.11.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with three hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.12 Support Vector Machines, Random Forest and Deep CNF with Four Hidden Layers

The twelfth ensemble combines support vector machine, random forest and deep convolutional neural field with four hidden layers. Tables 4.3.1.12.1 and 4.3.1.12.2 show the results of combining SVM, random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.6% on validation sets and 83.0% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	82.0	86.3	74.2	87.0
2	82.0	84.4	68.7	85.7
3	83.0	85.2	76.6	87.7
4	83.0	84.6	78.2	86.0
5	82.0	86.7	79.7	84.6
6	83.0	84.5	72.4	85.6
7	83.0	83.7	77.3	86.0
Overall Accuracy:	**82.6**	**85.0**	**75.3**	**86.0**

Table 4.3.1.12.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with four hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	PrecisionH	PrecisionE	PrecisionL
1	82.3	85.0	72.6	86.2	90.5	84.7	75.7
2	82.5	85.0	76.6	84.1	86.8	84.4	78.8
3	83.1	84.5	75.7	86.0	88.9	85.5	77.9
4	82.8	85.0	75.5	85.0	88.0	82.6	79.2
5	82.1	83.7	78.0	83.0	88.6	79.0	79.1
6	83.0	84.6	76.7	84.8	89.5	80.4	79.7
7	84.8	87.7	77.4	85.6	91.1	84.5	79.4
Overall Accuracy:	**83.0**	**85.2**	**76.0**	**85.0**	**89.2**	**82.9**	**78.6**

Table 4.3.1.12.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with four hidden layers on test sets of 7-fold cross-validation on CB513.

4.3.1.13 Support Vector Machines, Random Forest and Deep CNF with Five Hidden Layers

The thirteenth ensemble combines support vector machine, random forest and deep convolutional neural field with five hidden layers classifiers. Tables 4.3.1.13.1 and 4.3.1.13.2 show the results of combining SVM, random forest and deep CNF on validation and test sets of seven fold cross-validation experiment on CB513, respectively. The overall accuracy Q_3 is obtained as 82.6% on validation sets and 83.0% on test sets.

Fold Number	Q_3	Q_H	Q_E	Q_L
1	82.0	86.4	74.0	86.8
2	82.0	84.2	69.0	85.6
3	83.0	85.0	76.8	87.7
4	83.0	84.7	78.2	86.0
5	82.0	87.1	79.8	84.6

6	83.0	84.2	72.1	85.5
7	83.0	83.7	76.9	86.2
Overall Accuracy:	**82.6**	**85.0**	**75.3**	**86.0**

Table 4.3.1.13.1 Recall measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with five hidden layers on validation sets of CB513.

Fold Number	Q_3	Q_H	Q_E	Q_L	**PrecisionH**	**PrecisionE**	**PrecisionL**
1	82.4	85.5	72.5	86.1	90.4	84.8	75.8
2	82.5	85.5	76.8	83.8	86.5	84.3	79.1
3	83.3	84.8	75.6	86.3	89.3	85.6	78.1
4	82.7	85.1	76.0	84.4	87.5	82.3	79.5
5	81.9	84.0	77.7	82.5	88.6	78.2	79.0
6	83.0	85.0	76.7	84.5	89.3	79.9	79.9
7	84.7	87.6	77.2	85.6	90.9	84.7	79.2
Overall Accuracy:	**83.0**	**85.4**	**76.0**	**84.7**	**89.0**	**82.7**	**78.7**

Table 4.3.1.13.2 Recall and precision measures for each class type and overall accuracy of model averaging ensemble that combines SVM, random forest and deep CNF with five hidden layers on test sets of 7-fold cross-validation on CB513.

4.4 Comparison of Results

Tables 4.4.1 and 4.4.2 show the recall measures for all methods implemented in this thesis on validation and test sets of CB513, respectively. Based on these results the best accuracy is obtained when the SVM and deep CNF methods are combined. The accuracy of the ensemble that combines SVM, deep CNF and random forest is also comparable.

Method	Q_3	Q_H	Q_E	Q_L
SVM	**83.4**	85.2	75.7	85.8
RF	**82.0**	83.0	72.0	86.0
DeepCNF$_3$	**89.8**	92.3	87.5	89.3
DeepCNF$_4$	**90.0**	91.9	87.7	89.6
DeepCNF$_5$	**90.1**	92.1	88.2	89.1
DeepCNF$_{3,4,5}$	**90.2**	91.8	87.7	89.3
SVM+RF	**83.2**	84.6	74.5	86.5
SVM+DeepCNF$_{3,4,5}$	**83.5**	85.7	76.3	85.4
SVM+DeepCNF$_3$	**82.9**	85.0	74.4	85.6
SVM+DeepCNF$_4$	**83.5**	85.4	76.2	85.6
SVM+DeepCNF$_5$	**83.4**	85.6	76.3	85.3
RF+DeepCNF$_{3,4,5}$	**83.0**	85.0	74.6	85.8
RF+DeepCNF$_3$	**82.9**	85.0	74.4	85.6
RF+DeepCNF$_4$	**83.0**	84.8	74.5	86.0
RF+DeepCNF$_5$	**82.9**	85.0	74.6	85.6
SVM+RF+DeepCNF$_{3,4,5}$	**83.4**	85.0	75.3	86.0
SVM+RF+DeepCNF$_3$	**83.3**	85.2	75.0	86.0
SVM+RF+DeepCNF$_4$	**83.3**	85.0	75.3	86.0
SVM+RF+DeepCNF$_5$	**83.3**	85.0	75.3	86.0

Table 4.4.1 Recall measures for each class type and overall accuracy of all models on validation sets of CB513.

Method	Q_3	Q_H	Q_E	Q_L	PrecisionH	PrecisionE	PrecisionL
SVM	**82.8**	85.1	76.1	84.5	88.9	82.6	78.5
RF	**81.8**	83.2	73.7	85.0	89.2	81.9	76.7
DeepCNF$_3$	**82.6**	86.0	76.5	83.0	88.0	81.4	79.1

DeepCNF$_4$	**82.6**	85.8	76.5	83.3	88.0	82.0	78.9
DeepCNF$_5$	**82.6**	86.5	76.3	82.9	87.6	81.7	79.2
DeepCNF$_{3,4,5}$	**82.6**	86.1	76.2	83.3	87.9	81.8	79.0
SVM+RF	**82.8**	84.5	75.4	85.2	89.3	83.0	78.1
SVM+DeepCNF$_{3,4,5}$	**83.0**	85.7	76.5	84.3	88.7	82.6	79.0
SVM+DeepCNF$_3$	**83.0**	85.8	76.6	84.2	88.8	82.4	79.1
SVM+DeepCNF$_4$	**83.0**	85.6	76.5	84.4	88.8	82.8	78.9
SVM+DeepCNF$_5$	**83.0**	85.9	76.6	84.1	88.5	82.6	79.1
RF+DeepCNF$_{3,4,5}$	**82.8**	85.2	75.5	84.6	88.9	82.4	78.5
RF+DeepCNF$_3$	**82.7**	85.2	75.8	84.4	89.0	82.1	78.5
RF+DeepCNF$_4$	**82.8**	85.1	75.7	84.7	89.0	82.6	78.4
RF+DeepCNF$_5$	**82.7**	85.4	75.5	84.3	88.7	82.4	78.5
SVM+RF+DeepCNF$_{3,4,5}$	**83.0**	85.3	76.0	84.8	89.1	82.7	78.6
SVM+RF+DeepCNF$_3$	**83.0**	85.3	76.0	84.8	89.2	82.7	78.6
SVM+RF+DeepCNF$_4$	**83.0**	85.2	76.0	85.0	89.2	82.9	78.6
SVM+RF+DeepCNF$_5$	**83.0**	85.4	76.0	84.7	89.0	82.7	78.7

Table 4.4.2 Recall and precision measures for each class type and overall accuracy of all models on test sets of 7-fold cross-validation on CB513.

CHAPTER 5

CONCLUSION

In this thesis, we optimized a support vector machine, a deep CNF and a random forest classifier for protein secondary structure prediction. These are employed at the second stage of a hybrid method. We also analyzed the performance of an ensemble method that combines the predictions of the classifiers. When the individual classifiers are compared, the most accurate method is the support vector machine, followed by deep CNF and random forest. The ensemble methods achieve better accuracy rates although the improvement is small. Nonetheless the ensemble approach has the potential to improve the accuracy of secondary structure prediction. We are planning to implement and compare other ensemble techniques such as stacking as a future work.

In constructing structural profile matrices we used distant templates by setting the percentage of sequence identity threshold to 20% (i.e. removing templates having greater similarity than this threshold to query). Therefore our results are obtained in the most difficult setting. In previous studies that are conducted at similar difficulty levels approximately 80% - 83% accuracy has been obtained. In this study, we achieve nearly 83% accuracy by the ensemble method, which is comparable to state-of-the art. As a future work we are planning to repeat the optimization experiments for other difficulty levels and for dihedral angle class and solvent accessibility class predictions.

BIBLIOGRAPHY

[1] D. T. Jones, “Protein structure prediction in genomics,” *Briefings in Bioinformatics*, vol. 2, no. 2, pp. 111–125, 2001.

[2] J. Zeng, S. Zhu, H. Yan, “Towards accurate human promoter recognition: a review of currently used sequence features and classification methods,” *Briefings in Bioinformatics*, vol. 10, no. 5, pp. 498–508, 2009.

[3] G. Bologna, and R. D. Appel, “A comparison study on protein fold recognition,” *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, vol. 5, pp. 2492-2496, IEEE, Nov. 2002.

[4] A. Chinnasamy, W. K. Sung, and A. Mittal, “Protein structure and fold prediction using tree-augmented naive Bayesian classifier,” *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 04, pp. 803-819, 2005.

[5] C. H. Ding, and I. Dubchak, “Multi-class protein fold recognition using support vector machines and neural networks,” *Bioinformatics*, vol. 17, no. 4, pp. 349-358, 2001.

[6] A. Bhola, S. K. Yadav, and A. K. Tiwari, “Machine Learning based Approach for Protein function prediction using sequence derived properties,” *International journal of computer applications*, vol. 105, no. 12, 2014.

[7] B. E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d’Alche–Buc, “Gene networks inference using dynamic Bayesian networks,” *Bioinformatics*, vol. 19, no. 2, pp. ii138-ii148, 2003.

[8] J. M. Dale, L. Popescu, and P. D. Karp, “Machine learning methods for metabolic pathway prediction,” *BMC bioinformatics*, vol. 11, no. 1, pp. 15, 2010.

[9] Protein structure prediction: https://en.wikipedia.org/wiki/Protein_structure_prediction

[10] W. Kabsch, and C. Sander, “Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features,” *Biopolymers*, vol. 22, no. 12, pp. 2577-2637, 1983.

[11] F.C Bernstein, et al. “The protein data bank,” *The FEBS Journal*, vol. 80, no. 2, pp. 319:324, 1977.

[12] J. J. Ward, L. J. McGuffin, B. F. Buxton, and D. T. Jones, “Secondary structure prediction with support vector machines,” *Bioinformatics*, vol. 19, no. 13, pp. 1650-1655, 2003.

[13] S. Hua, and Z. Sun, “A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach,” *J. Mol. Biol.*, vol. 308, pp. 397–407, 2001.

[14] B. Rost, and C. Sander, *J. Mol. Biol.* , vol. 232, pp. 584–599, 1993.

[15] J. A. Cuff, and G. J. Barton, *Proteins: Struct. Funct. Genet.* , vol. 34, pp. 508–519,1999.

[16] H. Kim, H. Park, “Protein secondary structure prediction based on an improved support vector machines approach,” *Protein Engineering*, vol. 16, no. 8, pp. 553-560, 2003.

[17] J. Gubbi, D. T. Lai, M. Palaniswami and M. Parker, “Protein secondary structure prediction using support vector machines and a new feature representation,” *International Journal of Computational Intelligence and Applications*, vol. 6, no. 4, pp. 551-567, 2006.

[18] K. E. Chen, L. A. Kurgan, and J. Ruan, “Prediction of protein structural class using novel evolutionary collocation-based sequence representation,” *Journal of computational chemistry*, vol. 29, no. 10, pp. 1596-1604, 2008.

[19] T. G. Dietterich, “Ensemble methods in machine learning,” *International workshop on multiple classifier systems*, Springer Berlin Heidelberg, vol. 1857, pp. 1–15, June 2000.

[20] R. King, M. Ouali, A. Strong, A. Aly, A. Elmaghraby, M. Kantardzic, and D. Page, “Is it better to combine predictions?,” *Protein Engineering*, vol. 13, pp. 15-19, 2000.

[21] P. Kountouris, M. Agathocleous, V. J. Promponas, G. Christodoulou, S. Hadjicostas, V. Vassiliades, and C. Christodoulou, “A comparative study on filtering protein secondary structure prediction,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 9, no. 3, pp. 731-739, 2012.

[22] M. Alirezaee, A. Dehzangi, and E. Mansoori, “Ensemble of neural networks to solve class imbalance problem of protein secondary structure

prediction," *International Journal of Artificial Intelligence & Applications*, vol. 3, no. 6, pp. 9, 2012.

[23] G. Pollastri, P. Baldi, P. Fariselli, and R. Casadio, "Prediction of coordination number and relative solvent accessibility in proteins," *Proteins: Structure, Function, and Bioinformatics*, vol. 47, no. 2, pp. 142-153, 2002.

[24] W. Li, Y. Chen, and Y. Zhao, "Multi-layer ensemble classifiers on protein secondary structure prediction," *In International Conference on Intelligent Computing*, Springer Berlin Heidelberg, pp. 79-85, Sep. 2008.

[25] C. N. Magnan, and P. Baldi, "SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity," *Bioinformatics*, vol. 30, no. 18, pp. 2592-2597, 2014.

[26] 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: Structure, Function, and Bioinformatics*, vol. 47, no. 2, pp. 228-235, 2002.

[27] H. Bouziane, B. Messabih, and A. Chouarfia, "Effect of simple ensemble methods on protein secondary structure prediction," *Soft Computing*, vol. 19, no 6, pp. 1663-1678, 2015.

[28] A. Dehzangi, K. Paliwal, A. Sharma, O. Dehzangi, and A. Sattar, "A combination of feature extraction methods with an ensemble of different classifiers for protein structural class prediction problem," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 10, no. 30, pp. 564-575, 2013.

[29] J. A. Cuff, G. J. Barton, "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction," *Proteins: Structure, Function, and Bioinformatics*, vol. 34, no. 4, pp. 508-519, 1999.

[30] X. Q. Yao, H. Zhu, and Z. S. She, "A dynamic Bayesian network approach to protein secondary structure prediction," *BMC bioinformatics*, vol. 9, no. 1, pp. 49, 2008.

[31] Z. Aydin, A. Singh, J. Bilmes and W. S. Noble, (2011). "Learning sparse models for a dynamic Bayesian network classifier of protein secondary structure," *BMC bioinformatics*, vol. 12, no. 1, pp. 154, 2011.

[32] J. Peng, L. Bo, and J. Xu, "Conditional neural fields," *In Advances in neural information processing systems*, pp. 1419-1427, 2009.

[33] S. Wang, J. Peng, J. Ma, and J. Xu, “Protein secondary structure prediction using deep convolutional neural fields,” *Scientific reports 6*, 2016.

[34] Protein: https://en.wikipedia.org/wiki/Protein

[35] Protein yapısı: http://tr.wikipedia.org/wiki/Protein_yapısı

[36] Protein structure: https://en.wikipedia.org/wiki/Protein_structure

[37] F. M. Richards, and C. E. Kundrot, “Identification of structural motifs from protein coordinate data: secondary structure and first-level supersecondary structure,” *Proteins: Structure, Function, and Bioinformatics*, vol. 3, no. 2, pp. 71-84, 1988.

[38] D. Frishman, and P. Argos, “Knowledge-based protein secondary structure assignment,” *Proteins: Structure, Function, and Bioinformatics*, vol. 23, no. 4, pp. 566-579, 1995.

[39] A. Zemla, C. Venclovas, K. Fidelis, and B. Rost, “A modified definition of Sov, a segment-based measure for protein secondary structure prediction assessment,” *Proteins: Structure, Function, and Bioinformatics*, vol. 34, no. 2, pp. 220-223, 1999.

[40] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” *Biochim. Biophys. Acta*, vol. 405, pp. 442-451, 1975.

[41] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” Nucleic Acids Res., vol. 25, no. 17, pp. 3389-402, 1997.

[42] M. Remmert, A. Biegert, A. Hauser, J. Söding, “HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment,” *Nat. Methods*., vol. 9, no. 2, pp. 173-175, 2011.

[43] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, and G. Penn, “Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition,” *In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4277-4280, IEEE, March 2012.

[44] Z. Aydin, D. Baker, and W. S. Noble, “Constructing Structural Profiles for Protein Torsion Angle Prediction,” *In BIOINFORMATICS*, pp. 26-35, 2015.

[45] Z. Aydin, J. Thompson, J. Bilmes, D. Baker, and W. S. Noble, “Protein torsion angle class prediction by a hybrid architecture of Bayesian and neural networks,” *Proceedings of the International Conference on Bioinformatics & Computational Biology (BIOCOMP)*. The Steering Committee of The World Congress in Computer

Science, Computer Engineering and Applied Computing (WorldComp), pp.473, Jan. 2012.

[46] Z. Aydin, D. Baker, and W. S. Noble, (2015, January). "Template Scoring Methods for Protein Torsion Angle Prediction," *In International Joint Conference on Biomedical Engineering Systems and Technologies*, Springer, Cham., pp. 206-223, Jan. 2015.

[47] S. M. Reynolds, L. Käll, M. E. Riffle, J. A. Bilmes, and W. S. Noble, "Transmembrane topology and signal peptide prediction using dynamic bayesian networks," *PLoS Comput Biol*, vol. 4, no. 11, pp. e1000213, 2008.

[48] Graphical Models Toolkit (GMTK): http://melodi.ee.washington.edu/~bilmes/gmtk/.

[49] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine learning*, vol. 46, no. 1, pp. 131-159, 2002.

[50] S. Busuttil, "Support vector machines," *In Proceedings of CSAW'03*, pp. 34, Nov. 2003.

[51] Picture of non-linear SVM, Researchgate user atarina Moreira, https://www.researchgate.net/figure/260283043_fig13_Figure-A15-The-non-linear-SVM-classifier-with-the-kernel-trick

[52] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[53] H. M. Karakoyun, and M. Hacıbeoğlu, "Biyomedikal Veri Kümeleri İle Makine Öğrenmesi Sınıflandırma Algoritmalarının İstatistiksel Olarak Karşılaştırılması," *DEÜ Mühendislik Fakültesi Mühendislik Bilimleri Dergisi*, vol. 16, pp. 30-41, 2014.

[54] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench. In Intelligent Information Systems," 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on. IEEE, pp. 357-361, Dec. 1994.

[55] A. Statnikov, L. Wang, and C. F. Aliferis, "A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification," *BMC bioinformatics*, vol. 9, no. 1, pp. 319, 2008.

[56] B. J. Lee, M. S. Shin, Y. J. Oh, H. S. Oh, and K. H. Ryu, "Identification of protein functions using a machine-learning approach based on sequence-derived properties," *Proteome science*, vol. 7, no. 1, pp. 27, 2009.

[57] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[58] Y. Qi, M. Oja, J. Weston, and W. S. Noble, “A unified multitask architecture for predicting local protein properties,” *PloS one*, vol. 7, no. 3, pp. e32235, 2012.

[59] P. Di Lena, K. Nagata, and P. Baldi, “Deep architectures for protein contact map prediction,” *Bioinformatics*, vol. 28, no. 19, pp. 2449-2457, 2012.

[60] R. Collobert, and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” *In Proceedings of the 25th international conference on Machine learning*, pp. 160-167, ACM, July 2008.

[61] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *Journal of Machine Learning Research*, pp. 2493-2537, 12 Aug. 2011.

[62] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.

[63] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[64] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[65] Auc-maximized Deep Convolutional Neural Fields for Sequence Labeling - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/286263785_fig1_Figure-1-Illustration-of-a-DeepCNF-Here-i-is-the-position-index-and-X-i-the-associated [accessed 17 May, 2017]

[66] C. C. Chang and C. J. Lin, “LIBSVM : a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no.3, pp. 27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[67] S. Wang, Training Conditional Random Fields by Maximizing AUC, May 2015. Software available at https://github.com/realbigws/DeepCNF_AUC.